Chapter 8 Programming

Important!

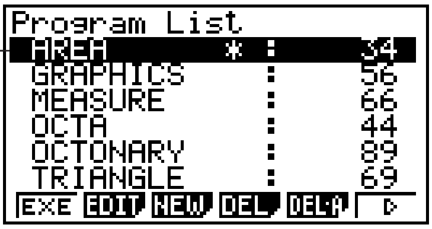
Input in the **PRGM** mode is always performed using the Linear input/output mode.

1. Basic Programming Steps

Commands and calculations are executed sequentially, just like manual calculation multi-statements.

1. From the Main Menu, enter the **PRGM** mode. When you do, a program list appears on the display.

Selected program area
(use ▲ and ▼ to move)



```
Program List
 AREA      * :   34
 GRAPHICS    :   56
 MEASURE     :   66
 OCTA        :   44
 OCTONARY    :   89
 TRIANGLE    :   69
```

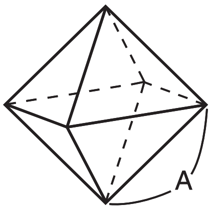
   Files are listed in the alphabetic sequence of their names.

2. Register a file name.
3. Input the program.
4. Run the program.

- The values to the right of the program list indicate the number of bytes used by each program.
- A file name can be up to eight characters long.
- The following are the characters you can use in a file name: A through Z, r, θ, spaces, [,], {, }, ’, ”, ~, 0 through 9, ., +, –, ×, ÷
- Registering a file name uses 32 bytes of memory.

Example **To calculate the surface area (cm²) and volume (cm³) of three regular octahedrons when the length of one side is 7, 10, and 15 cm, respectively**

Store the calculation formula under the file name OCTA.



The following are the formulas used for calculating surface area S and volume V of a regular octahedron for which the length of one side A is known.

$$S = 2\sqrt{3}\,A^2, \quad V = \frac{\sqrt{2}}{3}A^3$$

① MENU PRGM

② F3(NEW) 9(O) ln(C) ÷(T) X,θ,T(A) EXE

③ SHIFT VARS(PRGM) F4(?) → ALPHA X,θ,T(A) F6(▷) F5(:)

2 × SHIFT x²(√‾) 3 × ALPHA X,θ,T(A) x² F6(▷) F6(▷) F5(◢)

SHIFT x²(√‾) 2 ÷ 3 × ALPHA X,θ,T(A) ^ 3

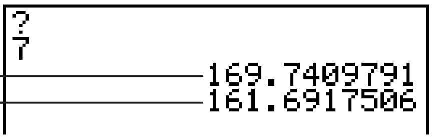
EXIT EXIT

④ F1(EXE) or EXE

7 EXE(Value of A)

EXE

S when A = 7
V when A = 7

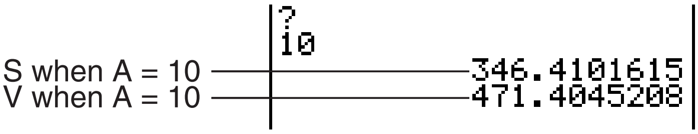


```
?
7
                169.7409791
                161.6917506
```

EXE EXE

1 0 EXE

EXE



```
?
10
                346.4101615
                471.4045208
```

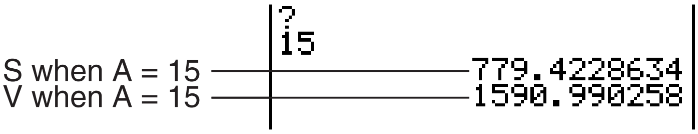
S when A = 10

V when A = 10

EXE EXE

1 5 EXE

EXE *1



```
?
15
                779.4228634
                1590.990258
```

S when A = 15

V when A = 15

*1 Pressing EXE while the program's final result is on the display exits the program.

- You can also run a program while in the **RUN•MAT** (or **RUN**) mode by inputting: Prog "<file name>" EXE.
- Pressing EXE while the final result of a program executed using this method is on the display re-executes the program.
- An error occurs if the program specified by Prog "<file name>" cannot be found.

2. PRGM Mode Function Keys

- {**NEW**} ... {new program}

• When you are registering a file name

- {**RUN**}/{**BASE**} ... {general calculation}/{number base} program input
- {**π0**} ... {password registration}
- {**SYBL**} ... {symbol menu}

• When you are inputting a program —— F1(RUN) … default

- {**TOP**}/{**BTM**} ... {top}/{bottom} of program
- {**SRC**} ... {search}
- {**MENU**} ... {mode menu}
  - {**STAT**}/{**MAT**}*/{**LIST**}/{**GRPH**}/{**DYNA**}*/{**TABL**}/{**RECR**}* ... {statistic}/{matrix}/{list}/{graph}/{Dynamic Graph}/{Table}/{recursion} menu
- {**A↔a**} ... {toggles between upper-case and lower-case input}
- {**CHAR**} ... {displays a screen for selecting various mathematical symbols, special symbols, and accented characters}

* Not included on the fx-7400GII

- Pressing SHIFT VARS (PRGM) displays the following program (PRGM) menu.
- {**COM**} ... {program command menu}
- {**CTL**} ... {program control command menu}
- {**JUMP**} ... {jump command menu}
- {**?**}/{**◢**} ... {input}/{output} command
- {**CLR**}/{**DISP**} ... {clear}/{display} command menu
- {**REL**} ... {conditional jump relational operator menu}

- {**I/O**} ... {I/O control/transfer command menu}
- {**:**} ... {multi-statement command}
- {**STR**} ... {string command}

See “Command Reference” on page 8-7 for full details on each of these commands.

- Pressing SHIFT MENU (SET UP) displays the mode command menu shown below.
  - {**ANGL**}/{**COOR**}/{**GRID**}/{**AXES**}/{**LABL**}/{**DISP**}/{**S/L**}/{**DRAW**}/{**DERV**}/{**BACK**}/{**FUNC**}/{**SIML**}/{**S-WIN**}/{**LIST**}/{**LOCS**}*/{**T-VAR**}/{**ΣDSP**}*/{**RESID**}/{**CPLX**}/{**FRAC**}/{**Y•SPD**}*/{**DATE**}*/{**PMT**}*/{**PRD**}*/{**INEQ**}/{**SIMP**}/{**Q1Q3**}   * Not included on the fx-7400GII

See “Setup Screen Function Key Menus” on page 1-27 for details about each of these commands.

● When you are inputting a program —— F2 (BASE)*1

- {**TOP**}/{**BTM**}/{**SRC**}
- {**MENU**}
  - {**d~o**} ... {decimal}/{hexadecimal}/{binary}/{octal} value input
  - {**LOG**} ... {bitwise operator}
  - {**DISP**} ... conversion of displayed value to {decimal}/{hexadecimal}/{binary}/{octal}
- {**A↔a**}/{**SYBL**}

- Pressing SHIFT VARS (PRGM) displays the following PRGM (PROGRAM) menu.
  - {**Prog**} ... {program recall}
  - {**JUMP**}/{**?**}/{**◢**}
  - {**REL**} ... {conditional jump relational operator menu}
  - {**:**} ... {multi-statement command}

- Pressing SHIFT MENU (SET UP) displays the mode command menu shown below.
  - {**Dec**}/{**Hex**}/{**Bin**}/{**Oct**}

*1 Programs input after pressing F2 (BASE) are indicated by B to the right of the file name.

- {**EXE**}/{**EDIT**} ... program {execute}/{edit}
- {**NEW**} ... {new program}
- {**DEL**}/{**DEL•A**} ... {specific program}/{all program} delete
- {**SRC**}/{**REN**} ... file name {search}/{change}

3. Editing Program Contents

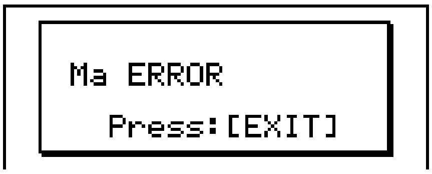
■ Debugging a Program

A problem in a program that keeps the program from running correctly is called a “bug”, and the process of eliminating such problems is called “debugging”. Either of the following symptoms indicates that your program contains bugs that require debugging.

- Error messages appearing when the program is run
- Results that are not within your expectations

• To eliminate bugs that cause error messages

An error message, like the one shown to the right, appears whenever something illegal occurs during program execution.



```
Ma ERROR
  Press:[EXIT]
```

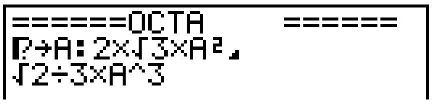
When such a message appears, press EXIT to display the place in the program where the error was caused. The cursor will be flashing at the location of the problem. Check the "Error Message Table" (page α-1) for steps you should take to correct the situation.

- Note that pressing EXIT does not display the location of the error if the program is password protected.

• To eliminate bugs that cause bad results

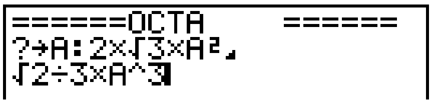
If your program produces results that are not what you normally expect, check the contents of the program and make necessary changes.

F1(TOP)... Moves the cursor to the top of the program



```
======OCTA    ======
?→A:2×√3×A²◢
√2÷3×A^3
```

F2(BTM)... Moves the cursor to the bottom of the program

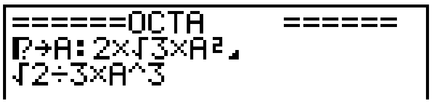


```
======OCTA    ======
?→A:2×√3×A²◢
√2÷3×A^3
```

■ Searching for Data Inside a Program

Example **To search for the letter "A" inside the program named OCTA**

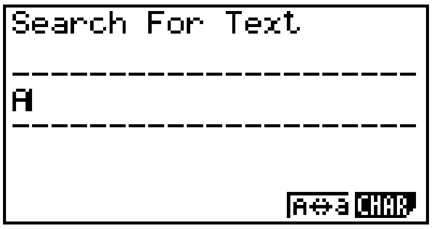
1. Recall the program.
2. Press F3(SRC) and input the data you want to find.



```
======OCTA    ======
?→A:2×√3×A²◢
√2÷3×A^3
```

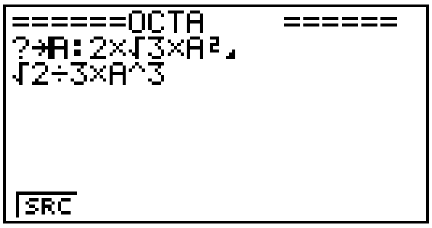
F3(SRC)

ALPHA X,θ,T(A)



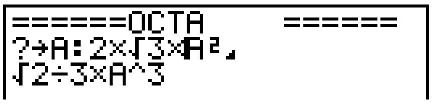
```
Search For Text
---------------------
A
---------------------
```

3. Press EXE to begin the search. The contents of the program appear on the screen with the cursor located at the first instance of the data you specified.*1



```
======OCTA    ======
?→A:2×√3×A²◢
√2÷3×A^3
```

4. Each press of EXE or F1(SRC) causes the cursor to jump to the next instance of the data you specified.*2



```
======OCTA    ======
?→A:2×√3×A²↵
√2÷3×A^3
```

*1 The message “Not Found” appears when the search data you specify cannot be found in the program.

*2 If there are no more instances of the data you specified, the search operation ends.

- You cannot specify the newline symbol (↵) or display command (◢) for the search data.
- Once the contents of the program are on the screen, you can use the cursor keys to move the cursor to another location before searching for the next instance of the data. Only the part of the program starting from the current cursor location is searched when you press EXE.
- Once the search finds an instance of your data, inputting characters or moving the cursor causes the search operation to be cancelled.
- If you make a mistake while inputting characters to search for, press AC to clear your input and re-input from the beginning.

4. File Management

■ Searching for a File

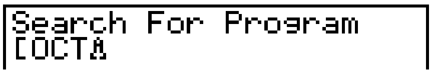
• To find a file using initial character search

Example To use initial character search to recall the program named OCTA

1. While the program list is on the display, press F6(▷)F1(SRC) and input the initial characters of the file you want to find.

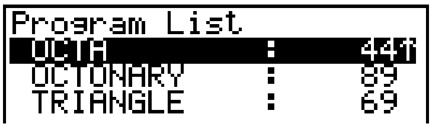
   F6(▷)F1(SRC)

   9(O)ln(C)÷(T)



```
Search For Program
[OCTA
```

2. Press EXE to search.
   - The name that starts with the characters you input highlights.



```
Program List
 OCTA        :    441
 OCTONARY    :     89
 TRIANGLE    :     69
```

- If there is no program whose file name starts with the characters you input, the message “Not Found” appears on the display. If this happens, press EXIT to clear the error message.

■ Editing a File Name

1. While the program list is on the display, use ▲ and ▼ to move the highlighting to the file whose name you want to edit and then press F6(▷)F2(REN).
2. Make any changes you want.
3. Press EXE to register the new name and return to the program list.

The program list is resorted according to the changes you made in the file name.

- If the modifications you make result in a file name that is identical to the name of a program already stored in memory, the message “Already Exists” appears. When this happens, you can perform either of the following two operations to correct the situation.

- Press EXIT to clear the error and return to the file name editing screen.
- Press AC to clear the input file name and input a new one.

■ Deleting a Program

• To delete a specific program

1. While the program list is on the display, use ▲ and ▼ to move the highlighting to the name of the program you want to delete.
2. Press F4(DEL).
3. Press F1(YES) to delete the selected program or F6(NO) to abort the operation without deleting anything.

• To delete all programs

1. While the program list is on the display, press F5(DEL•A).
2. Press F1(YES) to delete all the programs in the list or F6(NO) to abort the operation without deleting anything.

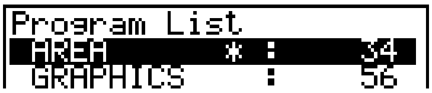
- You also can delete all programs by entering the **MEMORY** mode from the Main Menu. See "Chapter 11 Memory Manager" for details.

■ Registering a password

When inputting a program, you can protect it with a password that limits access to the program contents to those who know the password.

- You do not need to input the password to run a program.
- The password input procedure is identical to that used for file name input.

1. While the program list is on the display, press F3(NEW) and input the file name of the new program file.
2. Press F5(⊸O) and then input the password.
3. Press EXE to register the file name and password. Now you can input the contents of the program file.
4. After inputting the program, press SHIFT EXIT(QUIT) to exit the program file and return to the program list. Files that are password protected are indicated by an asterisk to the right of the file name.



```
Program List
 AREA       * :     34
 GRAPHICS     :     56
```

■ Recalling a Password Protected Program

1. In the program list, use ▲ and ▼ to move the highlighting to the name of the program you want to recall.
2. Press F2(EDIT).
3. Input the password and press EXE to recall the program.

- Inputting the wrong password when recalling a password protected program causes the message "Mismatch" to appear.

5. Command Reference

■ Command Index

The following are conventions that are used in this section when describing the various commands.

Boldface Text .... Actual commands and other items that always must be input are shown in boldface.

{Curly Brackets} .... Curly brackets are used to enclose a number of items, one of which must be selected when using a command. Do not input the curly brackets when inputting a command.

[Square Brackets] .... Square brackets are used to enclose items that are optional. Do not input the square brackets when inputting a command.

Numeric Expressions ... Numeric expressions (such as 10, 10 + 20, A) indicate constants, calculations, numeric constants, etc.

Alpha Characters .... Alpha characters indicate literal strings (such as AB).

Basic Operation Commands

? (Input Command)

Function: Prompts for input of values for assignment to variables during program execution.

Syntax: ? → <variable name>, "<prompt>" ? → <variable name>

Example: ? → A↵

Description:

- This command momentarily interrupts program execution and prompts for input of a value or expression for assignment to a variable. If you do not specify a prompt, execution of this command causes "?" to appear indicating the calculator is standing by for input. If a prompt is specified, "<prompt>?" appears to prompt input. Up to 255 bytes of text can be used for a prompt.
- Input in response to the input command must be a value or an expression, and the expression cannot be a multi-statement.
- You can specify a list name, matrix name, vector name, string memory, function memory (fn), graph (Yn), etc. as a variable name.

◢ (Output Command)

Function: Displays an intermediate result during program execution.

Description:

- This command momentarily interrupts program execution and displays alpha character text or the result of the calculation immediately before the command.
- The output command should be used at locations where you would normally press the EXE key during a manual calculation.

: (Multi-statement Command)

Function: Connects two statements for sequential execution without stopping.

Description:

- Unlike the output command (◢), statements connected with the multi-statement command are executed non-stop.
- The multi-statement command can be used to link two calculation expressions or two commands.
- You can also use a carriage return indicated by ↵ in place of the multi-statement command.

↵ (Carriage Return)

Function: Connects two statements for sequential execution without stopping.

Description:

- Operation of the carriage return is identical to that of the multi-statement command.
- You can create a blank line in a program by inputting a carriage return only. Using a carriage return in place of the multi-statement command makes the displayed program easier to read.

' (Comment Text Delimiter)

Function: Indicates comment text inserted inside a program.

Description: Inputting an apostrophe (') at the beginning of a line, causes everything from the beginning of the line up to the next Multi-statement Command (:), Carriage Return (↵), or Output Command (◢) is treated as comment text, which is ignored during execution.

Program Commands (COM)

If~Then~(Else~)IfEnd

Function: The Then-statement is executed only when the If-condition is true (non-zero). The Else-statement is executed when the If-condition is false (0). The IfEnd-statement is always executed following either the Then-statement or Else-statement.

Syntax:

$$\text{If}\quad \underset{\text{numeric expression}}{\underline{\langle\text{condition}\rangle}} \left\{\begin{array}{c}\hookleftarrow\\ :\\ \blacktriangle\end{array}\right\} \text{Then}\ \langle\text{statement}\rangle \left[\left\{\begin{array}{c}\hookleftarrow\\ :\\ \blacktriangle\end{array}\right\} \langle\text{statement}\rangle\right]$$

$$\left\{\begin{array}{c}\hookleftarrow\\ :\\ \blacktriangle\end{array}\right\} \left(\text{Else}\ \langle\text{statement}\rangle \left[\left\{\begin{array}{c}\hookleftarrow\\ :\\ \blacktriangle\end{array}\right\} \langle\text{statement}\rangle\right] \left\{\begin{array}{c}\hookleftarrow\\ :\\ \blacktriangle\end{array}\right\}\right) \text{IfEnd}$$

Parameters: condition, numeric expression

Description:

(1) If ~ Then ~ IfEnd

- When the condition is true, execution proceeds with the Then-statement and then continues with the statement following IfEnd.
- When the condition is false, execution jumps to the statement following IfEnd.

(2) If ~ Then ~ Else ~ IfEnd

- When the condition is true, execution proceeds with the Then-statement and then jumps to the statement following IfEnd.
- When the condition is false, execution jumps to the Else-statement and then continues with the statement following IfEnd.

For~To~(Step~)Next

Function: This command repeats everything between the For-statement and the Next-statement. The starting value is assigned to the control variable with the first execution, and the value of the control variable is changed according to the step value with each execution. Execution continues until the value of the control variable exceeds the ending value.

Syntax: For <starting value> → <control variable name> To <ending value>

$$\left(\text{Step}\ \langle\text{step value}\rangle\right) \left\{\begin{array}{c}\hookleftarrow\\ :\\ \blacktriangle\end{array}\right\} \text{Next}$$

Parameters:

- control variable name: A to Z
- starting value: value or expression that produces a value (i.e. sin x, A, etc.)
- ending value: value or expression that produces a value (i.e. sin x, A, etc.)
- step value: numeric value (default: 1)

Description:

- The default step value is 1.
- Making the starting value less than the ending value and specifying a positive step value causes the control variable to be incremented with each execution. Making the starting value greater than the ending value and specifying a negative step value causes the control variable to be decremented with each execution.

Do~LpWhile

Function: This command repeats specific commands as long as its condition is true (non-zero).

Syntax:

$$\text{Do} \begin{Bmatrix} \text{↵} \\ : \\ \text{◢} \end{Bmatrix} \text{<statement>} \begin{Bmatrix} \text{↵} \\ : \\ \text{◢} \end{Bmatrix} \text{LpWhile} \quad \underset{\text{numeric expression}}{\text{<}\underline{\text{condition}}\text{>}}$$

Parameters: expression

Description:

- This command repeats the commands contained in the loop as long as its condition is true (non-zero). When the condition becomes false (0), execution proceeds from the statement following the LpWhile-statement.
- Since the condition comes after the LpWhile-statement, the condition is tested (checked) after all of the commands inside the loop are executed.

While~WhileEnd

Function: This command repeats specific commands as long as its condition is true (non-zero).

Syntax:

$$\text{While} \quad \underset{\text{numeric expression}}{\text{<}\underline{\text{condition}}\text{>}} \begin{Bmatrix} \text{↵} \\ : \\ \text{◢} \end{Bmatrix} \text{<statement>} \begin{Bmatrix} \text{↵} \\ : \\ \text{◢} \end{Bmatrix} \text{WhileEnd}$$

Parameters: expression

Description:

- This command repeats the commands contained in the loop as long as its condition is true (non-zero). When the condition becomes false (0), execution proceeds from the statement following the WhileEnd-statement.
- Since the condition comes after the While-statement, the condition is tested (checked) before the commands inside the loop are executed.

■ Program Control Commands (CTL)

Break

Function: This command breaks execution of a loop and continues from the next command following the loop.

Syntax: Break ↵

Description:

- This command breaks execution of a loop and continues from the next command following the loop.
- This command can be used to break execution of a For-statement, Do-statement, and While-statement.

Prog

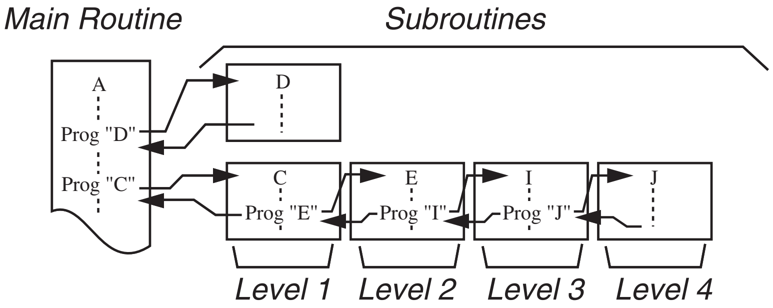
Function: This command specifies execution of another program as a subroutine. In the **RUN•MAT** (or **RUN**) mode, this command executes a new program.

Syntax: Prog "file name"↵

Example: Prog "ABC"↵

Description:

- Even when this command is located inside of a loop, its execution immediately breaks the loop and launches the subroutine.
- This command can be used as many times as necessary inside of a main routine to call up independent subroutines to perform specific tasks.
- A subroutine can be used in multiple locations in the same main routine, or it can be called up by any number of main routines.



- Calling up a subroutine causes it to be executed from the beginning. After execution of the subroutine is complete, execution returns to the main routine, continuing from the statement following the Prog command.
- A Goto~Lbl command inside of a subroutine is valid inside of that subroutine only. It cannot be used to jump to a label outside of the subroutine.
- If a subroutine with the file name specified by the Prog command does not exist, an error occurs.
- In the **RUN•MAT** (or **RUN**) mode, inputting the Prog command and pressing EXE launches the program specified by the command.

Return

Function: This command returns from a subroutine.

Syntax: Return↵

Description: Execution of the Return command inside a main routine causes execution of the program to stop. Execution of the Return command within a subroutine terminates the subroutine and returns to the program from which the subroutine was jumped to.

Stop

Function: This command terminates execution of a program.

Syntax: Stop↵

Description:

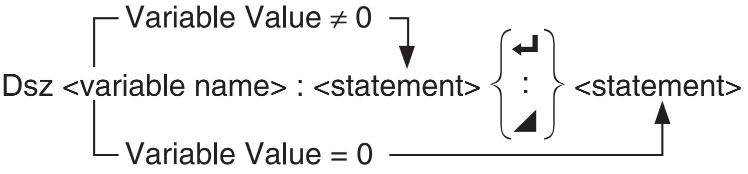
- This command terminates program execution.
- Execution of this command inside of a loop terminates program execution without an error being generated.

Jump Commands (JUMP)

Dsz

Function: This command is a count jump that decrements the value of a control variable by 1, and then jumps if the current value of the variable is zero.

Syntax:



Parameters: variable name: A to Z, r, θ

[Example] Dsz B : Decrements the value assigned to variable B by 1.

Description: This command decrements the value of a control variable by 1, and then tests (checks) it. If the current value is non-zero, execution continues with the next statement. If the current value is zero, execution jumps to the statement following the multi-statement command (:), display command (◢), or carriage return (↵).

Goto~Lbl

Function: This command performs an unconditional jump to a specified location.

Syntax: Goto <label name> ~ Lbl <label name>

Parameters: label name: value (0 to 9), variable (A to Z, r, θ)

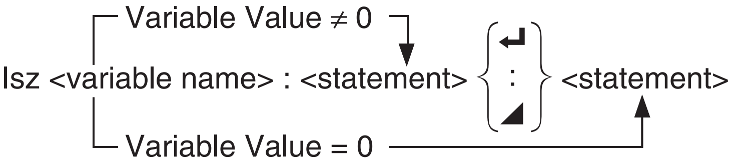
Description:

- This command consists of two parts: Goto n (where n is a parameter as described above) and Lbl n (where n is the parameter referenced by Goto n). This command causes program execution to jump to the Lbl-statement whose n parameter matches that specified by the Goto-statement.
- This command can be used to loop back to the beginning of a program or to jump to any location within the program.
- This command can be used in combination with conditional jumps and count jumps.
- If there is no Lbl-statement whose value matches that specified by the Goto-statement, an error occurs.

Isz

Function: This command is a count jump that increments the value of a control variable by 1, and then jumps if the current value of the variable is zero.

Syntax:



Parameters: variable name: A to Z, r, θ

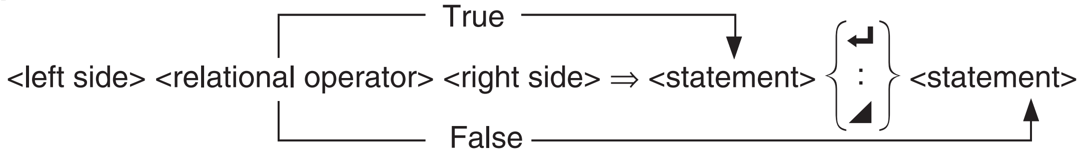
[Example] Isz A : Increments the value assigned to variable A by 1.

Description: This command increments the value of a control variable by 1, and then tests (checks) it. If the current value is non-zero, execution continues with the next statement. If the current value is zero, execution jumps to the statement following the multi-statement command (:), display command (◢), or carriage return (↵).

⇒ (Jump Code)

Function: This code is used to set up conditions for a conditional jump. The jump is executed whenever the conditions are false.

Syntax:



Parameters:

- left side/right side: variable (A to Z, r, θ), numeric constant, variable expression (such as: A × 2)
- relational operator: =, ≠, >, <, ≥, ≤ (page 8-18)

Description:

- The conditional jump compares the contents of two variables or the results of two expressions, and a decision is made whether or not to execute the jump based on the results of the comparison.
- If the comparison returns a true result, execution continues with the statement following the ⇒ command. If the comparison returns a false result, execution jumps to the statements following the multi-statement command (:), display command (◢), or carriage return (↵).

Menu

Function: Creates a branching menu in a program.

Syntax: Menu "<string (menu name)>", "<string (branch name) 1>", <value or variable 1>, "<string (branch name) 2>" ,<value or variable 2>, ... , "<string (branch name) *n*>", <value or variable *n*>

Parameters: value (0 to 9), variable (A to Z, r, θ)

Description:

- Each "<string (branch name) *n*>" ,<value or variable *n*> part is a branch set, and the entire branch set must be included.
- From two to nine branching sets can be included. An error occurs when there is only one or more than nine branching sets.
- Selecting a branch on the menu while the program is running jumps to the same type of label (Lbl *n*) as the one used in combination with the Goto command. Specifying ""OK", 3" for the ""<string (branch name) *n*>", <value or variable *n*>" part specifies a jump to Lbl 3.

Example:
```
Lbl 2↵
Menu "IS IT DONE?", "OK", 1, "EXIT", 2↵
Lbl 1↵
"IT'S DONE !"
```

■ Clear Commands (CLR)

ClrGraph

Function: This command clears the graph screen.

Syntax: ClrGraph↵

Description: This command clears the graph screen during program execution.

ClrList

Function: This command deletes list data.

Syntax: ClrList <list name>
ClrList

Parameters: list name: 1 to 26, Ans

Description: This command deletes the data in the list specified by “list name”. All list data is deleted if nothing is specified for “list name”.

ClrMat (Not included on the fx-7400GII)

Function: This command deletes matrix data.

Syntax: ClrMat <matrix name>
ClrMat

Parameters: matrix name: A to Z, Ans

Description: This command deletes the data in the matrix specified by “matrix name”. All matrix data is deleted if nothing is specified for “matrix name”.

ClrText

Function: This command clears the text screen.

Syntax: ClrText↵

Description: This command clears text from the screen during program execution.

ClrVct (Not included on the fx-7400GII/fx-9750GII)

Function: This command deletes vector data.

Syntax: ClrVct <vector name>
ClrVct

Parameters: vector name: A to Z, Ans

Description: This command deletes the data in the vector specified by “vector name”. All vector data is deleted if nothing is specified for “vector name”.

■ Display Commands (DISP)

DispF-Tbl, DispR-Tbl* * (Not included on the fx-7400GII) No parameters

Function: These commands display numeric tables.

Description:

- These commands generate numeric tables during program execution in accordance with conditions defined within the program.
- DispF-Tbl generates a function table, while DispR-Tbl generates a recursion table.

DrawDyna (Not included on the fx-7400GII) No parameters

Function: This command executes a Dynamic Graph draw operation.

Description: This command draws a Dynamic Graph during program execution in accordance with the drawing conditions defined within the program.

DrawFTG-Con, DrawFTG-Plt

No parameters

Function: This command uses values in a generated table to graph a function.

Description:

- This command draws a function graph in accordance with conditions defined within the program.
- DrawFTG-Con produces a connect type graph, while DrawFTG-Plt produces a plot type graph.

DrawGraph

No parameters

Function: This command draws a graph.

Description: This command draws a graph in accordance with the drawing conditions defined within the program.

DrawR-Con, DrawR-Plt

(Not included on the fx-7400GII) **No parameters**

Function: These commands graph recursion expressions, with a_n (b_n or c_n) as the vertical axis and n as the horizontal axis.

Description:

- These commands graph recursion expressions in accordance with conditions defined within the program, with a_n (b_n or c_n) as the vertical axis and n as the horizontal axis.
- DrawR-Con produces a connect type graph, while DrawR-Plt produces a plot type graph.

DrawRΣ-Con, DrawRΣ-Plt

(Not included on the fx-7400GII) **No parameters**

Function: These commands graph recursion expressions, with Σa_n (Σb_n or Σc_n) as the vertical axis and n as the horizontal axis.

Description:

- These commands graph recursion expressions in accordance with conditions defined within the program, with Σa_n (Σb_n or Σc_n) as the vertical axis and n as the horizontal axis.
- DrawRΣ-Con produces a connect type graph, while DrawRΣ-Plt produces a plot type graph.

DrawStat

Function: This draws a statistical graph.

Syntax: See "Using Statistical Calculations and Graphs in a Program" on page 8-25.

Description: This command draws a statistical graph in accordance with conditions defined within the program.

DrawWeb

(Not included on the fx-7400GII)

Function: This command graphs convergence/divergence of a recursion expression (WEB graph).

Syntax: DrawWeb <recursion type>[, <number of lines>] ↵

Example: DrawWeb a_{n+1} (b_{n+1} or c_{n+1}), 5 ↵

Description:

- This command graphs convergence/divergence of a recursion expression (WEB graph).
- Omitting the number of lines specification automatically specifies the default value 30.

PlotPhase (Not included on the fx-7400GII)

Function: Graphs a phase plot based on numeric sequences that correspond to the x-axis and y-axis.

Syntax: PlotPhase <x-axis numeric sequence name>, <y-axis numeric sequence name>

Description:

- Only the following commands can be input for each argument to specify the recursion table.

  a_n, b_n, c_n, a_{n+1}, b_{n+1}, c_{n+1}, a_{n+2}, b_{n+2}, c_{n+2}, Σa_n, Σb_n, Σc_n, Σa_{n+1}, Σb_{n+1}, Σc_{n+1}, Σa_{n+2}, Σb_{n+2}, Σc_{n+2}

- A memory ERROR occurs if you specify a numeric sequence name that does not have values stored in the recursion table.

Example: PlotPhase Σb_{n+1}, Σa_{n+1}

Graphs a phase plot using Σb_{n+1} for the x-axis and Σa_{n+1} for the y-axis.

■ Input/Output Commands (I/O)

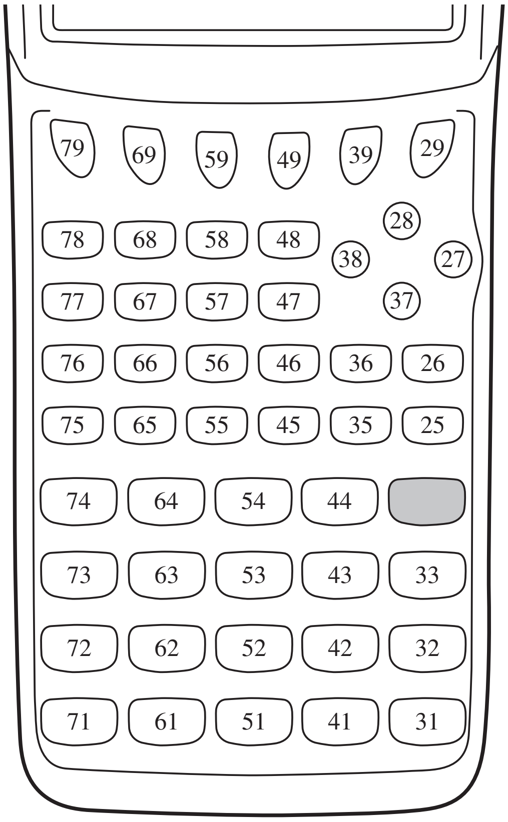
Getkey

Function: This command returns the code that corresponds to the last key pressed.

Syntax: Getkey ↵

Description:

- This command returns the code that corresponds to the last key pressed.



- A value of zero is returned if no key was pressed previous to executing this command.
- This command can be used inside of a loop.

Locate

Function: This command displays alpha-numeric characters at a specific location on the text screen.

Syntax: Locate <column number>, <line number>, <value>

Locate <column number>, <line number>, <numeric expression>

Locate <column number>, <line number>, "<string>"

[Example] Locate 1, 1, "AB"↵

Parameters:

- line number: number from 1 to 7
- column number: number from 1 to 21
- value and numeric expression
- string: character string

Description:

- This command displays values (including variable contents) or text at a specific location on the text screen. If there is a calculation input, that calculation result is displayed.
- The line is designated by a value from 1 to 7, while the column is designated by a value from 1 to 21.

(1, 1) →
← (21, 1)
(1, 7) →
← (21, 7)

Example: Cls↵

Locate 7, 1, "CASIO FX"

This program displays the text “CASIO FX” in the center of the screen.

- In some cases, the ClrText command should be executed before running the above program.

Receive(/ Send(

Function: This command receives data from and sends data to a connected device.

Syntax: Receive(<data>) / Send(<data>)

Description:

- This command receives data from and sends data to a connected device.
- The following types of data can be received (sent) by this command.
  - Individual values assigned to variables
  - Matrix data (all values - individual values cannot be specified)
  - List data (all values - individual values cannot be specified)

OpenComport38k / CloseComport38k

Function: Opens and closes the 3-pin COM port (serial).

Description: See the Receive38k/Send38k command below.

Receive38k / Send38k

Function: Executes data send and receive at a data rate of 38 kbps.

Syntax: Send38k <expression>

$$\text{Receive38k} \begin{Bmatrix} \text{<variable name>} \\ \text{<list name>} \end{Bmatrix}$$

Description:

- The OpenComport38k command must be executed before this command is executed.
- The CloseComport38k command must be executed after this command is executed.
- If this command is executed when the communication cable is not connected, program execution will continue without generating an error.

■ Conditional Jump Relational Operators (REL)

=, ≠, >, <, ≥, ≤

Function: These relational operators are used in combination with the conditional jump command.

Syntax: <left side> <relational operator> <right side>

Parameters:

- left side/right side: variable (A to Z, r, θ), numeric constant, variable expression (such as: A × 2)
- relational operator: =, ≠, >, <, ≥, ≤

■ Strings

A string is a series of characters enclosed in double quotes. In a program, strings are used to specify display text. A string made up of numbers (like "123") or an expression (like "x–1") cannot be processed as a calculation.

To display a string at a specific location on the screen, use the Locate command (page 8-17).

- To include double quotes (") or a backslash (\) in a string, put a backslash (\) in front of the double quotes (") or backslash (\).

  Example 1: To include Japan: “Tokyo” in a string
  "Japan:\"Tokyo\""

  Example 2: To include main\abc in a string
  "main\\abc"

You can input a backslash from the menu that appears when you press F6(CHAR) F2(SYBL) in the **PRGM** mode, or from the String category of the catalog that appears when you press SHIFT 4(CATALOG).

- You can assign strings to string memory (Str 1 through Str 20). For details about strings, see “String Memory” (page 2-7).
- You can use the “+” command (page 8-20) to connect strings inside of an argument.
- A function or command within a string function (Exp(, StrCmp(, etc.) is treated as a single character. For example, the “sin” function is treated as a single character.

Exp(

Function: Converts a string to an expression, and executes the expression.

Syntax: Exp("<string>"[)]

Exp►Str(

Function: Converts a graph expression to a string and assigns it to the specified variable.

Syntax: Exp►Str(<formula>, <string variable name>[)]

Description: A graph expression ($\mathrm{Y_n}$, r, $\mathrm{X_t}$, $\mathrm{Y_t}$, X), recursion formula (a_n, a_{n+1}, a_{n+2}, b_n, b_{n+1}, b_{n+2}, c_n, c_{n+1}, c_{n+2}), or function memory ($\mathrm{f_n}$) can be used as the first argument (<formula>).

StrCmp(

Function: Compares "<string 1>" and "<string 2>" (character code comparison).

Syntax: StrCmp("<string 1>", "<string 2>"[)]

Description: Compares two strings and returns one of the following values.

Returns 0 when "<string 1>" = "<string 2>".

Returns 1 when "<string 1>" > "<string 2>".

Returns –1 when "<string 1>" < "<string 2>".

StrInv(

Function: Inverts the sequence of a string.

Syntax: StrInv("<string>"[)]

StrJoin(

Function: Joins "<string 1>" and "<string 2>".

Syntax: StrJoin("<string 1>", "<string 2>"[)]

Note: The same result also can be achieved using the "+" command (page 8-20).

StrLeft(

Function: Copies a string up to the nth character from the left.

Syntax: StrLeft("<string>", n[)]   ($0 \leqq n \leqq 9999$, n is a natural number)

StrLen(

Function: Returns the length of a string (the number of its characters).

Syntax: StrLen("<string>"[)]

StrLwr(

Function: Converts all the characters of a string to lower case.

Syntax: StrLwr("<string>"[)]

StrMid(

Function: Extracts from the n-th to the m-th character of a string.

Syntax: StrMid("<string>", n [,m)]   (0 ≦ n ≦ 9999, n is a natural number)

Description: Omitting "m" will extract from the n-th character to the end of the string.

StrRight(

Function: Copies a string up to the nth character from the right.

Syntax: StrRight("<string>", n[)]   (0 ≦ n ≦ 9999, n is a natural number)

StrRotate(

Function: Rotates the left side part and right side part of a string at the nth character.

Syntax: StrRotate("<string>", [,n)]   (–9999 ≦ n ≦ 9999, n is an integer)

Description: Rotation is to the left when "n" is positive, and to the right when "n" is negative. Omitting "n" uses a default value of +1.

Example: StrRotate("abcde", 2) .... Returns the string "cdeab".

StrShift(

Function: Shifts a string left or right n characters.

Syntax: StrShift("<string>", [,n)]   (–9999 ≦ n ≦ 9999, n is an integer)

Description: Shift is to the left when "n" is positive, and to the right when "n" is negative. Omitting "n" uses a default value of +1.

Example: StrShift("abcde", 2) .... Returns the string "cde".

StrSrc(

Function: Searches "<string 1>" starting from the specified point (nth character from beginning of string) to determine if it contains the data specified by "<string 2>". If the data is found, this command returns the location of the first character of "<string 2>", starting from the beginning of "<string 1>".

Syntax: StrSrc("<string 1>", "<string 2>"[,n)]   (0 ≦ n ≦ 9999, n is a natural number)

Description: Omitting the start point causes the search to start from the beginning of "<string 1>".

StrUpr(

Function: Converts all the characters of a string to upper case.

Syntax: StrUpr("<string>"[)]

+

Function: Joins "<string 1>" and "<string 2>".

Syntax: "<string 1>"+"<string 2>"

Example: "abc"+"de"→Str 1.......... Assigns "abcde" to Str 1.

■ Other

RclCapt

Function: Displayed the contents specified by the capture memory number.

Syntax: RclCapt <capture memory number> (capture memory number: 1 to 20)

6. Using Calculator Functions in Programs

■ Text Display

You can include text in a program by simply enclosing it between double quotation marks. Such text appears on the display during program execution, which means you can add labels to input prompts and results.

Program	**Display**
"CASIO"	CASIO
? → X	?
"X =" ? → X	X = ?

- If the text is followed by a calculation formula, be sure to insert a display command (◢) between the text and calculation.
- Inputting more than 21 characters causes the text to move down to the next line. The screen scrolls automatically if the text exceeds 21 characters.
- You can specify up to 255 bytes of text for a comment.

■ Using Matrix Row Operations in a Program (Not available on the fx-7400GII)

These commands let you manipulate the rows of a matrix in a program.

- For this program, enter the **RUN•MAT** mode and then use the Matrix Editor to input the matrix, and then enter the **PRGM** mode to input the program.

• To swap the contents of two rows (Swap)

Example 1 **To swap the values of Row 2 and Row 3 in the following matrix:**

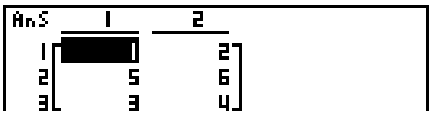
$$\textbf{Matrix A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

The following is the syntax to use for this program.

$$\text{Swap } \underbrace{A}_{\textit{Matrix name}},\ \underbrace{2,\ 3}_{\textit{Rows to be swapped}} \hookleftarrow$$

Mat A

Executing this program produces the following result.



```
AnS     1      2
  1 [   1      2 ]
  2 [   5      6 ]
  3 [   3      4 ]
```

• To calculate a scalar multiplication (∗ Row)

Example 2 **To calculate the product of Row 2 of the matrix in Example 1 and the scalar 4**

The following is the syntax to use for this program.

∗Row 4, A, 2↵

Row

Matrix name

Multiplier

Mat A

• To calculate a scalar multiplication and add the results to another row (∗Row+)

Example 3 **To calculate the product of Row 2 of the matrix in Example 1 and the scalar 4, then add the result to row 3**

The following is the syntax to use for this program.

∗Row+ 4, A, 2, 3↵

Rows to be added

Row for which scalar multiplication is to be calculated

Matrix name

Multiplier

Mat A

• To add two rows (Row+)

Example 4 **To add Row 2 to Row 3 of the matrix in Example 1**

The following is the syntax to use for this program.

Row+ A, 2, 3↵

Row number to be added to

Row number to be added

Matrix name

Mat A

■ Using Graph Functions in a Program

You can incorporate graph functions into a program to draw complex graphs and to overlay graphs on top of each other. The following shows various types of syntax you need to use when programming with graph functions.

- V-Window View Window –5, 5, 1, –5, 5, 1↵
- Graph function input Y = Type↵ ....Specifies graph type.

"X^2 – 3" → Y1[*1]↵

- Graph draw operation DrawGraph↵

*1 Input this Y1 with VARS F4(GRPH) F1(Y) 1 (displayed as Y1). A Syntax ERROR will occur if you input "Y" with the calculator keys.

● Syntax of other graphing functions

- V-Window View Window <Xmin>, <Xmax>, <Xscale>, <Ymin>, <Ymax>, <Yscale>, <Tθ min>, <Tθ max>, <Tθ pitch>

  StoV-Win <area of V-Win>............. area: 1 to 6

  RclV-Win <area of V-Win>............. area: 1 to 6

- Zoom Factor <X factor>, <Y factor>

  ZoomAuto...................................... Non-parameter

- Pict StoPict <$\underset{\text{numeric expression}}{\underline{\text{area of picture}}}$>................ area: 1 to 6

  RclPict <$\underset{\text{numeric expression}}{\underline{\text{area of picture}}}$> .... area: 1 to 6

- Sketch PlotOn <X-coordinate>, <Y-coordinate>

  PlotOff <X-coordinate>, <Y-coordinate>

  PlotChg <X-coordinate>, <Y-coordinate>

  PxlOn <line number>, <column number>

  PxlOff <line number>, <column number>

  PxlChg <line number>, <column number>

  PxlTest <line number>, <column number>

  Text <line number>, <column number>, "<text>"

  Text <line number>, <column number>, <expression>

  SketchThick <Sketch or Graph statement>

  SketchBroken <Sketch or Graph statement>

  SketchDot <Sketch or Graph statement>

  SketchNormal <Sketch or Graph statement>

  Tangent <function>, <X-coordinate>

  Normal <function>, <X-coordinate>

  Inverse <function>

  Line

  F-Line <X-coordinate 1>, <Y-coordinate 1>, <X-coordinate 2>, <Y-coordinate 2>

  Circle <center point X-coordinate>, <center point Y-coordinate>, <radius R value>

  Vertical <X-coordinate>

  Horizontal <Y-coordinate>

■ Using Dynamic Graph Functions in a Program

Using Dynamic Graph functions in a program makes it possible to perform repeated Dynamic Graph operations. The following shows how to specify the Dynamic Graph range inside a program.

- Dynamic Graph range

  1 → D Start↵

  5 → D End↵

  1 → D pitch↵

■ Using Table & Graph Functions in a Program

Table & Graph functions in a program can generate numeric tables and perform graphing operations. The following shows various types of syntax you need to use when programming with Table & Graph functions.

- Table range setting

  1 → F Start↵

  5 → F End↵

  1 → F pitch↵

- Numeric table generation

  DispF-Tbl↵

- Graph draw operation

  Connect type: DrawFTG-Con↵

  Plot type: DrawFTG-Plt↵

■ Using Recursion Table & Graph Functions in a Program

Incorporating Recursion Table & Graph functions in a program lets you generate numeric tables and perform graphing operations. The following shows various types of syntax you need to use when programming with Recursion Table & Graph functions.

- Recursion formula input

  a_{n+1} Type↵ .... Specifies recursion type.

  "3a_n + 2" → a_{n+1}↵

  "4b_n + 6" → b_{n+1}↵

- Table range setting

  1 → R Start↵

  5 → R End↵

  1 → a_0↵

  2 → b_0↵

  1 → a_n Start↵

  3 → b_n Start↵

- Numeric table generation

  DispR-Tbl↵

- Graph draw operation

  Connect type: DrawR-Con↵, DrawRΣ-Con↵

  Plot type: DrawR-Plt↵, DrawRΣ-Plt↵

- Statistical convergence/divergence graph (WEB graph)

  DrawWeb a_{n+1}, 10↵

■ Using List Sort Functions in a Program

These functions let you sort data in lists into ascending or descending order.

- Ascending order

```
①      ②
SortA (List 1, List 2, List 3)
        └──────Lists to be sorted (up to six can be specified)

① F4 F3 F1     ② OPTN F1 F1
```

- Descending order

```
③
SortD (List 1, List 2, List 3)
```

Lists to be sorted (up to six can be specified)

③ F4 F3 F2

Using Statistical Calculations and Graphs in a Program

Including statistical calculations and graphing operations in a program lets you calculate and graph statistical data.

• To set conditions and draw a statistical graph

Following a StatGraph command ("S-Gph1", "S-Gph2", or "S-Gph3"), you must specify the following graph conditions:

- Graph draw/non-draw status (DrawOn/DrawOff)
- Graph Type
- x-axis data location (list name)
- y-axis data location (list name)
- Frequency data location (list name)
- Mark Type
- Pie graph display setting (% or Data)
- Pie graph percentage data storage list specification (None or list name)
- First bar graph data (list name)
- Second and third bar graph data (list name)
- Bar graph orientation (Length or Horizontal)

The graph conditions that are required depends on the graph type. See "Changing Graph Parameters" (page 6-1).

- The following is a typical graph condition specification for a scatter diagram or xyLine graph.

  S-Gph1 DrawOn, Scatter, List 1, List 2, 1, Square ↵

  In the case of an xy line graph, replace "Scatter" in the above specification with "xyLine".

- The following is a typical graph condition specification for a normal probability plot.

  S-Gph1 DrawOn, NPPlot, List 1, Square ↵

- The following is a typical graph condition specification for a single-variable graph.

  S-Gph1 DrawOn, Hist, List 1, List 2 ↵

  The same format can be used for the following types of graphs, by simply replacing "Hist" in the above specification with the applicable graph type.

Histogram	Hist	Normal Distribution	N-Dist
Median Box	MedBox*1	Broken Line	Broken

*1 Outliers:On
S-Gph1 DrawOn, MedBox, List 1, 1, 1

Outliers:Off
S-Gph1 DrawOn, MedBox, List 1, 1, 0

- The following is a typical graph condition specification for a regression graph.

  S-Gph1 DrawOn, Linear, List 1, List 2, List 3 ↵

  The same format can be used for the following types of graphs, by simply replacing “Linear” in the above specification with the applicable graph type.

Linear Regression	Linear	Logarithmic Regression	Log
Med-Med.........................	Med-Med	Exponential Regression	ExpReg(a·e^bx)
Quadratic Regression	Quad		ExpReg(a·b^x)
Cubic Regression	Cubic	Power Regression	Power
Quartic Regression	Quart		

- The following is a typical graph condition specification for a sinusoidal regression graph.

  S-Gph1 DrawOn, Sinusoidal, List 1, List 2 ↵

- The following is a typical graph condition specification for a logistic regression graph.

  S-Gph1 DrawOn, Logistic, List 1, List 2 ↵

- The following is a typical graph condition specification for a pie graph.

  S-Gph1 DrawOn, Pie, List 1, %, None ↵

- The following is a typical graph condition specification for a bar graph.

  S-Gph1 DrawOn, Bar, List 1, None, None, StickLength ↵

- To draw a statistical graph, insert the “DrawStat” command following the graph condition specification line.

  ClrGraph

  S-Wind Auto

  {1, 2, 3} → List 1

  {1, 2, 3} → List 2

  S-Gph1 DrawOn, Scatter, List 1, List 2, 1, Square ↵

  DrawStat

■ Using Distribution Graphs in a Program (Not available on the fx-7400GII)

Special commands are used to draw distribution graphs in a program.

• To draw a normal cumulative distribution graph

① DrawDistNorm <Lower>, <Upper> [,σ, μ]

Population mean[*1]

Population standard deviation[*1]

Data upper limit

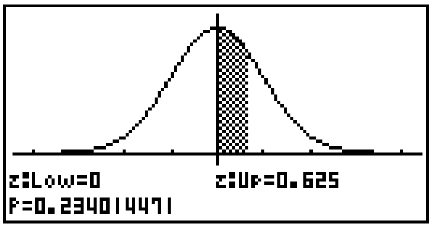
Data lower limit

① F4 F1 F5 F1

[*1] This can be omitted. Omitting these items performs the calculation using σ = 1 and μ = 0.

$$p = \frac{1}{\sqrt{2\pi}\sigma}\int_{Lower}^{Upper} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\,dx \qquad \text{ZLow} = \frac{Lower - \mu}{\sigma} \qquad \text{ZUp} = \frac{Upper - \mu}{\sigma}$$

- Executing DrawDistNorm performs the above calculation in accordance with the specified conditions and draws the graph. At this time the ZLow ≦ x ≦ ZUp region on the graph is filled in.



z:Low=0 z:Up=0.625
p=0.234014471

- At the same time, the p, ZLow, and ZUp calculation result values are assigned respectively to variables p, ZLow, and ZUp, and p is assigned to Ans.

• To draw a Student-*t* cumulative distribution graph

①DrawDistT <Lower>, <Upper>, <*df*>

Degree of freedom
Data upper limit
Data lower limit

① F4 F1 F5 F2

$$p = \int_{Lower}^{Upper} \frac{\Gamma\left(\frac{df+1}{2}\right)}{\Gamma\left(\frac{df}{2}\right)} \times \frac{\left(1+\frac{x^2}{df}\right)^{-\frac{df+1}{2}}}{\sqrt{\pi \times df}}\,dx \qquad \text{tLow} = Lower \qquad \text{tUp} = Upper$$

- Executing DrawDistT performs the above calculation in accordance with the specified conditions and draws the graph. At this time the Lower ≦ x ≦ Upper region on the graph is filled in.
- At the same time, the p calculation result value and the Lower and Upper input values are assigned respectively to variables p, tLow, and tUp, and p is assigned to Ans.

• To draw a χ^2 cumulative distribution graph

①DrawDistChi <Lower>, <Upper>, <*df*>

Degree of freedom
Data upper limit
Data lower limit

① F4 F1 F5 F3

$$p = \int_{Lower}^{Upper} \frac{1}{\Gamma\left(\frac{df}{2}\right)} \times \left(\frac{1}{2}\right)^{\frac{df}{2}} \times x^{\left(\frac{df}{2}-1\right)} \times e^{-\frac{x}{2}}\,dx$$

- Executing DrawDistChi performs the above calculation in accordance with the specified conditions and draws the graph. At this time the Lower ≦ x ≦ Upper region on the graph is filled in.
- At the same time, calculation result p is assigned to variables p and Ans.

• To draw an *F* cumulative distribution graph

①DrawDistF <Lower>, <Upper>, <*ndf*>, <*ddf*>

Degrees of freedom of denominator
Degrees of freedom of numerator
Data upper limit
Data lower limit

① F4 F1 F5 F4

$$p = \int_{Lower}^{Upper} \frac{\Gamma\left(\frac{ndf + ddf}{2}\right)}{\Gamma\left(\frac{ndf}{2}\right) \times \Gamma\left(\frac{ddf}{2}\right)} \times \left(\frac{ndf}{ddf}\right)^{\frac{ndf}{2}} \times x^{\left(\frac{ndf}{2} - 1\right)} \times \left(1 + \frac{ndf \times x}{ddf}\right)^{-\frac{ndf + ddf}{2}} dx$$

- Executing DrawDistF performs the above calculation in accordance with the specified conditions and draws the graph. At this time the Lower ≦ x ≦ Upper region on the graph is filled in.
- At the same time, calculation result p is assigned to variables p and Ans.

■ Performing Statistical Calculations in a Program

- Single-variable statistical calculation

①1-Variable List1, List 2

List 2 — *Frequency data (Frequency)*

List1 — *x-axis data (XList)*

① F4 F1 F6 F1

- Paired-variable statistical calculation

①2-Variable List 1, List 2, List 3

List 3 — *Frequency data (Frequency)*

List 2 — *y-axis data (YList)*

List 1 — *x-axis data (XList)*

① F4 F1 F6 F2

- Regression statistical calculation

①LinearReg(ax+b) List 1, List 2, List 3

LinearReg(ax+b) — *Calculation type**

List 3 — *Frequency data (Frequency)*

List 2 — *y-axis data (YList)*

List 1 — *x-axis data (XList)*

① F4 F1 F6 F6 F1 F1

* Any one of the following can be specified as the calculation type.

LinearReg(ax+b)	linear regression (ax+b type)
LinearReg(a+bx)	linear regression (a+bx type)
Med-MedLine	Med-Med calculation
QuadReg	quadratic regression
CubicReg	cubic regression
QuartReg	quartic regression
LogReg	logarithmic regression
ExpReg(a·e^bx)	exponential regression ($a{\cdot}e^{bx}$ type)
ExpReg(a·b^x)	exponential regression ($a{\cdot}b^x$ type)
PowerReg	power regression

- Sinusoidal regression statistical calculation

  SinReg List 1, List 2

  y-axis data (YList)

  x-axis data (XList)

- Logistic regression statistical calculation

  LogisticReg List 1, List 2

  y-axis data (YList)

  x-axis data (XList)

Performing Distribution Calculations in a Program

(Not available on the fx-7400GII)

- The following values are substituted whenever any of the values enclosed in brackets ([ ]) are omitted.

  σ=1, μ=0, tail=L (Left)

- For the calculation formula of each probability density function, see “Statistic Formula” (page 6-55).

• Normal Distribution

NormPD(: Returns the normal probability density (p value) for the specified data.

Syntax: NormPD(x[, σ, μ)]

- A single value or a list can be specified for x. Calculation result p is assigned to variables p and Ans (ListAns when x is a list).

NormCD(: Returns the normal cumulative distribution (p value) for the specified data.

Syntax: NormCD(Lower, Upper[, σ, μ)]

- Single values or lists can be specified for Lower and Upper. Calculation results p, ZLow, and ZUp are assigned respectively to variables p, ZLow, and ZUp. Calculation result p also is assigned to Ans (ListAns when Lower and Upper are lists).

InvNormCD(: Returns the inverse normal cumulative distribution (lower and/or upper value(s)) for the specified p value.

Syntax: InvNormCD(["L(or –1) or R(or 1) or C(or 0)",]p[,σ, μ])

tail (Left, Right, Central)

- A single value or a list can be specified for p. Calculation results are output in accordance with the tail setting as described below.

  tail = Left

  The Upper value is assigned to variables x1InvN and Ans (ListAns when p is a list).

  tail = Right

  The Lower value is assigned to variables x1InvN and Ans (ListAns when p is a list).

  tail = Central

  The Lower and Upper values are assigned respectively to variables x1InvN and x2InvN. Lower only is assigned to Ans (ListAns when p is a list).

• Student-*t* Distribution

tPD(: Returns the Student-*t* probability density (*p* value) for the specified data.

Syntax: tPD(*x*, *df* [)]

- A single value or a list can be specified for *x*. Calculation result *p* is assigned to variables *p* and Ans (ListAns when *x* is a list).

tCD(: Returns the Student-*t* cumulative distribution (*p* value) for the specified data.

Syntax: tCD(Lower,Upper,*df* [)]

- Single values or lists can be specified for Lower and Upper. Calculation results *p*, tLow, and tUp are assigned respectively to variables *p*, tLow, and tUp. Calculation result *p* also is assigned to Ans (ListAns when Lower and Upper are lists).

InvTCD(: Returns the inverse Student-*t* cumulative distribution (Lower value) for the specified *p* value.

Syntax: InvTCD(*p*,*df* [)]

- A single value or a list can be specified for *p*. The Lower value is assigned to the *x*Inv and Ans variables (ListAns when *p* is a list).

• χ^2 Distribution

ChiPD(: Returns the χ^2 probability density (*p* value) for the specified data.

Syntax: ChiPD(*x*,*df* [)]

- A single value or a list can be specified for *x*. Calculation result *p* is assigned to variables *p* and Ans (ListAns when *x* is a list).

ChiCD(: Returns the χ^2 cumulative distribution (*p* value) for the specified data.

Syntax: ChiCD(Lower,Upper,*df* [)]

- Single values or lists can be specified for Lower and Upper. Calculation result *p* is assigned to variables *p* and Ans (ListAns when Lower and Upper are lists).

InvChiCD(: Returns the inverse χ^2 cumulative distribution (Lower value) for the specified *p* value.

Syntax: InvChiCD(*p*,*df* [)]

- A single value or a list can be specified for *p*. The Lower value is assigned to the *x*Inv and Ans variables (ListAns when *p* is a list).

• *F* Distribution

FPD(: Returns the *F* probability density (*p* value) for the specified data.

Syntax: FPD(*x*,*ndf*,*ddf* [)]

- A single value or a list can be specified for *x*. Calculation result *p* is assigned to variables *p* and Ans (ListAns when *x* is a list).

FCD(: Returns the *F* cumulative distribution (*p* value) for the specified data.

Syntax: FCD(Lower,Upper,*ndf*,*ddf* [)]

- Single values or lists can be specified for Lower and Upper. Calculation result *p* is assigned to variables *p* and Ans (ListAns when Lower and Upper are lists).

InvFCD(: Returns the inverse *F* cumulative distribution (Lower value) for the specified data.

Syntax: InvFCD(*p*,*ndf*,*ddf* [)]

- A single value or a list can be specified for *p*. The Lower value is assigned to the *x*Inv and Ans variables (ListAns when *p* is a list).

• Binomial Distribution

BinomialPD(: Returns the binomial probability (*p* value) for the specified data.

Syntax: BinomialPD([*x*,]*n*,P[)]

- A single value or a list can be specified for *x*. Calculation result *p* is assigned to variables *p* and Ans (ListAns when *x* is a list).

BinomialCD(: Returns the binomial cumulative distribution (*p* value) for the specified data.

Syntax: BinomialCD([X,]*n*,P[)]

- A single value or a list can be specified for each X. Calculation result *p* is assigned to variables *p* and Ans (ListAns when X is omitted or is a list).

InvBinomialCD(: Returns the inverse binomial cumulative distribution for the specified data.

Syntax: InvBinomialCD(*p*,*n*,P[)]

- A single value or a list can be specified for *p*. The calculation result X value is assigned to the *x*Inv and Ans variables (ListAns when *p* is a list).

• Poisson Distribution

PoissonPD(: Returns the Poisson probability (*p* value) for the specified data.

Syntax: PoissonPD(*x*, *μ*[)]

- A single value or a list can be specified for *x*. Calculation result *p* is assigned to variables *p* and Ans (ListAns when *x* is a list).

PoissonCD(: Returns the Poisson cumulative distribution (*p* value) for the specified data.

Syntax: PoissonCD(X,*μ*[)]

- A single value or a list can be specified for each X. Calculation result *p* is assigned to variables *p* and Ans (ListAns when X is a list).

InvPoissonCD(: Returns the inverse Poisson cumulative distribution for the specified data.

Syntax: InvPoissonCD(*p*,*μ*[)]

- A single value or a list can be specified for *p*. The calculation result X value is assigned to the *x*Inv and Ans variables (ListAns when *p* is a list).

• Geometric Distribution

GeoPD(: Returns the geometric probability (*p* value) for the specified data.

Syntax: GeoPD(*x*, P[)]

- A single value or a list can be specified for *x*. Calculation result *p* is assigned to variables *p* and Ans (ListAns when *x* is a list).

GeoCD(: Returns the geometric cumulative distribution (p value) for the specified data.

Syntax: GeoCD(X,P[)]

- A single value or a list can be specified for each X. Calculation result p is assigned to variables p and Ans (ListAns when X is a list).

InvGeoCD(: Returns the inverse geometric cumulative distribution for the specified data.

Syntax: InvGeoCD(p,P[)]

- A single value or a list can be specified for p. The calculation result X value is assigned to the xInv and Ans variables (ListAns when p is a list).

• Hypergeometric Distribution

HypergeoPD(: Returns the hypergeometric probability (p value) for the specified data.

Syntax: HypergeoPD(x, n, M, N[)]

- A single value or a list can be specified for x. Calculation result p is assigned to variables p and Ans (ListAns when x is a list).

HypergeoCD(: Returns the hypergeometric cumulative distribution (p value) for the specified data.

Syntax: HypergeoCD(X, n, M, N[)]

- A single value or a list can be specified for each X. Calculation result p is assigned to variables p and Ans (ListAns when X is a list).

InvHypergeoCD(: Returns the inverse hypergeometric cumulative distribution for the specified data.

Syntax: InvHypergeoCD(p, n, M, N[)]

- A single value or a list can be specified for p. The calculation result X value is assigned to the xInv and Ans variables (ListAns when p is a list).

■ Using the TEST Command to Execute a Command in a Program

(Not available on the fx-7400GII)

- The following are the specifications ranges for the "μ condition" argument of the command.

  "<" or –1 when $\mu < \mu_0$

  "≠" or 0 when $\mu \neq \mu_0$

  ">" or 1 when $\mu > \mu_0$

  The above also apply for the "ρ condition" and "β&ρ condition" specification methods.
- For explanations of arguments that are not covered in detail here, see "Tests" (page 6-23) and "Input and Output Terms of Tests, Confidence Interval, and Distribution" (page 6-52).
- For the calculation formula of each command, see "Statistic Formula" (page 6-55).

• Z Test

OneSample$Z$Test: Executes 1-sample Z-test calculation.

Syntax: OneSample$Z$Test "μ condition", μ_0, σ, $\bar{x}$, n

Output Values: Z, p, $\bar{x}$, n are assigned respectively to variables z, p, $\bar{x}$, n and to ListAns elements 1 through 4.

Syntax: OneSample$Z$Test "μ condition", μ_0, σ, List[, Freq]

Output Values: Z, p, $\bar{x}$, s_x, n are assigned respectively to variables z, p, $\bar{x}$, s_x, n and to ListAns elements 1 through 5.

TwoSample$Z$Test: Executes 2-sample Z-test calculation.

Syntax: TwoSample$Z$Test "μ_1 condition", σ_1, σ_2, $\bar{x}_1$, n_1, $\bar{x}_2$, n_2

Output Values: Z, p, $\bar{x}_1$, $\bar{x}_2$, n_1, n_2 are assigned respectively to variables z, p, $\bar{x}_1$, $\bar{x}_2$, n_1, n_2 and to ListAns elements 1 through 6.

Syntax: TwoSample$Z$Test "μ_1 condition", σ_1, σ_2, List1, List2[, Freq1 [, Freq2]]

Output Values: Z, p, $\bar{x}_1$, $\bar{x}_2$, s_{x1}, s_{x2}, n_1, n_2 are assigned respectively to variables z, p, $\bar{x}_1$, $\bar{x}_2$, s_{x1}, s_{x2}, n_1, n_2 and to ListAns elements 1 through 8.

OneProp$Z$Test: Executes 1-proportion Z-test calculation.

Syntax: OneProp$Z$Test "p condition", p_0, x, n

Output Values: Z, p, $\hat{p}$, n are assigned respectively to variables z, p, $\hat{p}$, n and to ListAns elements 1 through 4.

TwoProp$Z$Test: Executes 2-proportion Z-test calculation.

Syntax: TwoProp$Z$Test "p_1 condition", x_1, n_1, x_2, n_2

Output Values: Z, p, $\hat{p}_1$, $\hat{p}_2$, $\hat{p}$, n_1, n_2 are assigned respectively to variables z, p, $\hat{p}_1$, $\hat{p}_2$, $\hat{p}$, n_1, n_2 and to ListAns elements 1 through 7.

• *t* Test

OneSampleTTest: Executes 1-sample t-test calculation.

Syntax: OneSampleTTest "μ condition", μ_0, $\bar{x}$, s_x, n
OneSampleTTest "μ condition", μ_0, List[, Freq]

Output Values: t, p, $\bar{x}$, s_x, n are assigned respectively to the variables with the same names and to ListAns elements 1 through 5.

TwoSampleTTest: Executes 2-sample t-test calculation.

Syntax : TwoSampleTTest "μ_1 condition", $\bar{x}_1$, s_{x1}, n_1, $\bar{x}_2$, s_{x2}, n_2[,Pooled condition]
TwoSampleTTest "μ_1 condition", List1, List2, [, Freq1[, Freq2[, Pooled condition]]]

Output Values: When Pooled condition = 0, t, p, df, $\bar{x}_1$ $\bar{x}_2$, s_{x1}, s_{x2}, n_1, n_2 are assigned respectively to the variables with the same names and to ListAns elements 1 through 9.
When Pooled condition = 1, t, p, df, $\bar{x}_1$, $\bar{x}_2$, s_{x1}, s_{x2}, s_p, n_1, n_2 are assigned respectively to the variables with the same names and to ListAns elements 1 through 10.

Note: Specify 0 when you want to turn off the Pooled condition and 1 when you want to turn it on. Omitting the input is treated as Pooled condition off.

LinRegTTest: Executes linear regression t-test calculation.

Syntax: LinRegTTest "β&ρ condition", XList, YList[, Freq]

Output Values: t, p, df, a, b, s, r, r^2 are assigned respectively to the variables with the same names and to ListAns elements 1 through 8.

• χ^2 Test

ChiGOFTest: Executes a chi-square goodness of fit test.

Syntax: ChiGOFTest List 1, List 2, df, List 3
(List 1 is the Observed list, List 2 is the Expected list, and List 3 is the CNTRB list.)

Output Values: χ^2, p, df are assigned respectively to the variables with the same names and to ListAns elements 1 through 3. The CNTRB list is stored in List 3.

ChiTest: Executes a chi-square test.

Syntax: ChiTest MatA, MatB
(MatA is the Observed matrix and MatB is the Expected matrix.)

Output Values: χ^2, p, df are assigned respectively to the variables with the same names and to ListAns elements 1 through 3. The Expected matrix is assigned to MatB.

• F Test

TwoSample*F*Test: Executes 2-sample F-test calculation.

Syntax: TwoSample*F*Test "σ_1 condition", s_{x1}, n_1, s_{x2}, n_2

Output Values: F, p, s_{x1}, s_{x2}, n_1, n_2 are assigned respectively to the variables with the same names and to ListAns elements 1 through 6.

Syntax: TwoSample*F*Test "σ_1 condition", List1, List2, [, Freq1 [, Freq2]]

Output Values: F, p, $\bar{x}_1$, $\bar{x}_2$, s_{x1}, s_{x2}, n_1, n_2 are assigned respectively to the variables with the same names and to ListAns elements 1 through 8.

• ANOVA

OneWayANOVA: Executes one-factor ANOVA analysis of variance.

Syntax: OneWayANOVA List1, List2
(List1 is Factor list (A) and List2 is the Dependent list.)

Output Values: Adf, Ass, Ams, AF, Ap, ERRdf, ERRss, ERRms are assigned respectively to variables Adf, SSa, MSa, Fa, pa, Edf, SSe, MSe.

Also, output values are assigned to MatAns as shown below.

$$\text{MatAns} = \begin{bmatrix} Adf & Ass & Ams & AF & Ap \\ ERRdf & ERRss & ERRms & 0 & 0 \end{bmatrix}$$

TwoWayANOVA: Executes two-factor ANOVA analysis of variance.

Syntax: TwoWayANOVA List1, List2, List3 (List1 is Factor list (A), List2 is Factor list (B), and List3 is the Dependent list.)

Output Values: Adf, Ass, Ams, AF, Ap, Bdf, Bss, Bms, BF, Bp, ABdf, ABss, ABms, ABF, ABp, ERRdf, ERRss, ERRms are assigned respectively to variables Adf, SSa, MSa, Fa, pa, Bdf, SSb, MSb, Fb, pb, ABdf, SSab, MSab, Fab, pab, Edf, SSe, MSe.

Also, output values are assigned to MatAns as shown below.

$$\text{MatAns} = \begin{bmatrix} Adf & Ass & Ams & AF & Ap \\ Bdf & Bss & Bms & BF & Bp \\ ABdf & ABss & ABms & ABF & ABp \\ ERRdf & ERRss & ERRms & 0 & 0 \end{bmatrix}$$

■ Performing Financial Calculations in a Program

(Not available on the fx-7400GII)

• Setup Commands

- Date Mode Setting for Financial Calculations

  DateMode365....... 365 days

  DateMode360....... 360 days

- Payment Period Setting

  PmtBgn................. Start of period

  PmtEnd................. End of period

- Bond Calculation Payment Periods

  PeriodsAnnual...... Annual

  PeriodsSemi......... Semiannual

• Financial Calculation Commands

For the meaning of each argument, see "Chapter 7 Financial Calculation (TVM)".

• Simple Interest

Smpl_SI: Returns the interest based on simple interest calculation.

Syntax: Smpl_SI(n, I%, PV)

Smpl_SFV: Returns the total of principal and interest based on simple interest calculation.

Syntax: Smpl_SFV(n, I%, PV)

• Compound Interest

Note:

- P/Y and C/Y can be omitted for all compound interest calculations. When they are omitted, calculations are performed using P/Y=12 and C/Y=12.
- If you perform a calculation that uses a compound interest function (Cmpd_n(, Cmpd_I%(, Cmpd_PV(, Cmpd_PMT(, Cmpd_FV(), the argument(s) you input and the calculation results will be saved to the applicable variables (n, I%, PV, etc.). If you perform a calculation that uses any other type of financial calculation function, the argument and calculation results are not assigned to variables.

Cmpd_n: Returns the number of compound periods.

Syntax: Cmpd_n(I%, PV, PMT, FV, P/Y, C/Y)

Cmpd_I%: Returns the annual interest.

Syntax: Cmpd_I%(n, PV, PMT, FV, P/Y, C/Y)

Cmpd_PV: Returns the present value (loan amount for installment payments, principal for savings).

Syntax: Cmpd_PV(n, I%, PMT, FV, P/Y, C/Y)

Cmpd_PMT: Returns equal input/output values (payment amounts for installment payments, deposit amounts for savings) for a fixed period.

Syntax: Cmpd_PMT(*n*, *I*%, PV, FV, P/Y, C/Y)

Cmpd_FV: Returns the final input/output amount or total principal and interest.
Syntax: Cmpd_FV(*n*, *I*%, PV, PMT, P/Y, C/Y)

- **Cash Flow (Investment Appraisal)**

Cash_NPV: Returns the net present value.
Syntax: Cash_NPV(*I*%, Csh)

Cash_IRR: Returns the internal rate of return.
Syntax: Cash_IRR(Csh)

Cash_PBP: Returns the payback period.
Syntax: Cash_PBP(*I*%, Csh)

Cash_NFV: Returns the net future value.
Syntax: Cash_NFV(*I*%, Csh)

- **Amortization**

Amt_BAL: Returns the remaining principal balance following payment PM2.
Syntax: Amt_BAL(PM1, PM2, *I*%, PV, PMT, P/Y, C/Y)

Amt_INT: Returns the interest paid for payment PM1.
Syntax: Amt_INT(PM1, PM2, *I*%, PV, PMT, P/Y, C/Y)

Amt_PRN: Returns the principal and interest paid for payment PM1.
Syntax: Amt_PRN(PM1, PM2, *I*%, PV, PMT, P/Y, C/Y)

Amt_ΣINT: Returns the total principal and interest paid from payment PM1 to PM2.
Syntax: Amt_ΣINT(PM1, PM2, *I*%, PV, PMT, P/Y, C/Y)

Amt_ΣPRN: Returns the total principal paid from payment PM1 to PM2.
Syntax: Amt_ΣPRN(PM1, PM2, *I*%, PV, PMT, P/Y, C/Y)

- **Interest Rate Conversion**

Cnvt_EFF: Returns the interest rate converted from the nominal interest rate to the effective interest rate.
Syntax: Cnvt_EFF(*n*, *I*%)

Cnvt_APR: Returns the interest rate converted from the effective interest rate to the nominal interest rate.
Syntax: Cnvt_APR(*n*, *I*%)

- **Cost, Selling Price, Margin Calculations**

Cost: Returns the cost based on a specified selling price and margin.
Syntax: Cost(Sell, Margin)

Sell: Returns the selling price based on a specified cost and margin.
Syntax: Sell(Cost, Margin)

Margin: Returns the margin based on a specified cost and selling price.
Syntax: Margin(Cost, Sell)

- **Day/Date Calculations**

Days_Prd: Returns the number of days from a specified d1 to specified d2.

Syntax: Days_Prd(MM1, DD1, YYYY1, MM2, DD2, YYYY2)

- **Bond Calculations**

Bond_PRC: Returns in list form bond prices based on specified conditions.

Syntax: Bond_PRC(MM1, DD1, YYYY1, MM2, DD2, YYYY2, RDV, CPN, YLD) = {PRC, INT, CST}

Bond_YLD: Returns the yield based on specified conditions.

Syntax: Bond_YLD(MM1, DD1, YYYY1, MM2, DD2, YYYY2, RDV, CPN, PRC)

7. PRGM Mode Command List

Not all of the commands listed below are available on all models covered by this manual.

RUN Program

F4 (MENU) key			
Level 1	**Level 2**	**Level 3**	**Command**
STAT	DRAW	On	**DrawOn**
		Off	**DrawOff**
	GRPH	GPH1	**S-Gph1_**
		GPH2	**S-Gph2_**
		GPH3	**S-Gph3_**
		Scat	**Scatter**
		xy	**xyLine**
		Hist	**Hist**
		Box	**MedBox**
		Bar	**Bar**
		N-Dis	**N-Dist**
		Brkn	**Broken**
		X	**Linear**
		Med	**Med-Med**
		X^2	**Quad**
		X^3	**Cubic**
		X^4	**Quart**
		Log	**Log**
		*1	
		Pwr	**Power**
		Sin	**Sinusoidal**
		NPP	**NPPlot**
		Lgst	**Logistic**
		Pie	**Pie**
	List		**List_**
	TYPE	*2	
	DIST	DrwN	**DrawDistNorm_**
		Drwt	**DrawDistT_**
		DrwC	**DrawDistChi_**
		DrwF	**DrawDistF_**
	CALC	1VAR	**1-Variable_**
		2VAR	**2-Variable_**
		*3	
		Med	**Med-MedLine_**
		X^2	**QuadReg_**
		X^3	**CubicReg_**
		X^4	**QuartReg_**
		Log	**LogReg_**
		*4	
		Pwr	**PowerReg_**
		Sin	**SinReg_**
		Lgst	**LogisticReg_**
MAT	Swap		**Swap_**
	×Rw		**∗Row_**
	×Rw+		**∗Row+_**
	Rw+		**Row+_**
LIST	Srt-A		**SortA(**
	Srt-D		**SortD(**
GRPH	SEL	On	**G_SelOn_**
		Off	**G_SelOff_**
	TYPE	Y=	**Y=Type**
		r=	**r=Type**
		Parm	**ParamType**
		X=	**X=Type**
		Y>	**Y>Type**
		Y<	**Y<Type**
		Y≥	**Y≥Type**
		Y≤	**Y≤Type**
		X>	**X>Type**
		X<	**X<Type**
		X≥	**X≥Type**
		X≤	**X≤Type**
	STYL	—	**NormalG_**
		━	**ThickG_**
		·····	**BrokenThickG_**
		······	**DotG_**
	GMEM	Sto	**StoGMEM_**
		Rcl	**RclGMEM_**
DYNA	On		**D_SelOn_**
	Off		**D_SelOff_**
	Var		**D_Var_**
	TYPE	Y=	**Y=Type**
		r=	**r=Type**
		Parm	**ParamType**
TABL	On		**T_SelOn_**
	Off		**T_SelOff_**
	TYPE	Y=	**Y=Type**
		r=	**r=Type**
		Parm	**ParamType**
	STYL	—	**NormalG_**
		━	**ThickG_**
		·····	**BrokenThickG_**
		······	**DotG_**
RECR	SEL+S	On	**R_SelOn_**
		Off	**R_SelOff_**
		—	**NormalG_**
		━	**ThickG_**
		·····	**BrokenThickG_**
		······	**DotG_**
	TYPE	a_n	**a_nType**
		a_{n+1}	**a_{n+1}Type**
		a_{n+2}	**a_{n+2}Type**
	n.a_n··	n	**n**
		a_n	**a_n**
		a_{n+1}	**a_{n+1}**
		a_{n+2}	**a_{n+2}**
		b_n	**b_n**
		b_{n+1}	**b_{n+1}**
		b_{n+2}	**b_{n+2}**
		c_n	**c_n**
		c_{n+1}	**c_{n+1}**
		c_{n+2}	**c_{n+2}**
		Σa_n	**Σa_n**
		Σa_{n+1}	**Σa_{n+1}**
		Σa_{n+2}	**Σa_{n+2}**
		Σb_n	**Σb_n**
		Σb_{n+1}	**Σb_{n+1}**
		Σb_{n+2}	**Σb_{n+2}**
		Σc_n	**Σc_n**
		Σc_{n+1}	**Σc_{n+1}**
		Σc_{n+2}	**Σc_{n+2}**

	RANG	a_0	**Sel_a_0**
		a_1	**Sel_a_1**

OPTN key

Level 1	Level 2	Level 3	Command
LIST	List		**List_**
	L→M		**List→Mat(**
	Dim		**Dim_**
	Fill		**Fill(**
	Seq		**Seq(**
	Min		**Min(**
	Max		**Max(**
	Mean		**Mean(**
	Med		**Median(**
	Aug		**Augment(**
	Sum		**Sum_**
	Prod		**Prod_**
	Cuml		**Cuml_**
	%		**Percent_**
	⊿		**⊿List_**
MAT	Mat		**Mat_**
	M→L		**Mat→List(**
	Det		**Det_**
	Trn		**Trn_**
	Aug		**Augment(**
	Iden		**Identity_**
	Dim		**Dim_**
	Fill		**Fill(**
	Ref		**Ref_**
	Rref		**Rref_**
	Vct		**Vct_**
	DotP		**DotP(**
	CrsP		**CrossP(**
	Angle		**Angle(**
	UntV		**UnitV(**
	Norm		**Norm(**
CPLX	i		**i**
	Abs		**Abs_**
	Arg		**Arg_**
	Conj		**Conjg_**
	ReP		**ReP_**
	ImP		**ImP_**
	▶r∠θ		**▶r∠θ**
	▶a+bi		**▶a+bi**
CALC	Solve		**Solve(**
	d/dx		**d/dx(**
	d^2/dx^2		**d^2/dx^2(**
	∫dx		**∫(**
	SolveN		**SolveN(**
	FMin		**FMin(**
	FMax		**FMax(**
	Σ(		**Σ(**
	log$_a$b		**log$_a$b(**
	Int÷		**Int÷**
	Rmdr		**Rmdr**
	Simp		**▶Simp**
STAT	$\hat{x}$		**$\hat{x}$**
	$\hat{y}$		**$\hat{y}$**
	DIST	*5	
	S·Dev		**StdDev(**
	Var		**Variance(**
	TEST	*6	
CONV	▶		**▶**
	LENG	fm	**[fm]**
		Å	**[Å]**
		μm	**[μm]**
		mm	**[mm]**
		cm	**[cm]**
		m	**[m]**
		km	**[km]**
		AU	**[AU]**
		l.y.	**[l.y.]**
		pc	**[pc]**
		Mil	**[Mil]**
		in	**[in]**
		ft	**[ft]**
		yd	**[yd]**
		fath	**[fath]**
		rd	**[rd]**
		mile	**[mile]**
		n mile	**[n mile]**
	AREA	cm^2	**[cm^2]**
		m^2	**[m^2]**
		ha	**[ha]**
		km^2	**[km^2]**
		in^2	**[in^2]**
		ft^2	**[ft^2]**
		yd^2	**[yd^2]**
		acre	**[acre]**
		mile^2	**[mile^2]**
	VLUM	cm^3	**[cm^3]**
		mL	**[mL]**
		L	**[L]**
		m^3	**[m^3]**
		in^3	**[in^3]**
		ft^3	**[ft^3]**
		fl_oz(UK)	**[fl_oz(UK)]**
		fl_oz(US)	**[fl_oz(US)]**
		gal(US)	**[gal(US)]**
		gal(UK)	**[gal(UK)]**
		pt	**[pt]**
		qt	**[qt]**
		tsp	**[tsp]**
		tbsp	**[tbsp]**
		cup	**[cup]**
	TIME	ns	**[ns]**
		μs	**[μs]**
		ms	**[ms]**
		s	**[s]**
		min	**[min]**
		h	**[h]**
		day	**[day]**
		week	**[week]**
		yr	**[yr]**
		s-yr	**[s-yr]**
		t-yr	**[t-yr]**
	TMPR	°C	**[°C]**
		K	**[K]**
		°F	**[°F]**
		°R	**[°R]**
	VELO	m/s	**[m/s]**
		km/h	**[km/h]**
		knot	**[knot]**
		ft/s	**[ft/s]**
		mile/h	**[mile/h]**
	MASS	u	**[u]**
		mg	**[mg]**
		g	**[g]**
		kg	**[kg]**
		mton	**[mton]**
		oz	**[oz]**
		lb	**[lb]**
		slug	**[slug]**
		ton(short)	**[ton(short)]**
		ton(long)	**[ton(long)]**
	FORC	N	**[N]**
		lbf	**[lbf]**
		tonf	**[tonf]**
		dyne	**[dyne]**
		kgf	**[kgf]**
	PRES	Pa	**[Pa]**
		kPa	**[kPa]**
		mmH$_2$O	**[mmH$_2$O]**
		mmHg	**[mmHg]**
		atm	**[atm]**
		inH$_2$O	**[inH$_2$O]**
		inHg	**[inHg]**
		lbf/in^2	**[lbf/in^2]**
		bar	**[bar]**
		kgf/cm^2	**[kgf/cm^2]**
	ENGY	eV	**[eV]**
		J	**[J]**
		cal$_{th}$	**[cal$_{th}$]**
		cal$_{15}$	**[cal$_{15}$]**
		cal$_{IT}$	**[cal$_{IT}$]**
		kcal$_{th}$	**[kcal$_{th}$]**
		kcal$_{15}$	**[kcal$_{15}$]**
		kcal$_{IT}$	**[kcal$_{IT}$]**
		l-atm	**[l-atm]**
		kW•h	**[kW•h]**
		ft•lbf	**[ft•lbf]**
		Btu	**[Btu]**
		erg	**[erg]**
		kgf•m	**[kgf•m]**
	PWR	W	**[W]**
		cal$_{th}$/s	**[cal$_{th}$/s]**
		hp	**[hp]**
		ft•lbf/s	**[ft•lbf/s]**
		Btu/min	**[Btu/min]**
HYP	sinh		**sinh_**
	cosh		**cosh_**
	tanh		**tanh_**
	sinh^{-1}		**sinh^{-1}_**
	cosh^{-1}		**cosh^{-1}_**
	tanh^{-1}		**tanh^{-1}_**
PROB	X!		**!**
	nPr		**P**
	nCr		**C**
	RAND	Ran#	**Ran#_**
		Int	**RanInt#(**
		Norm	**RanNorm#(**

		Bin	**RanBin#(**
		List	**RanList#(**
	P(		**P(**
	Q(		**Q(**
	R(		**R(**
	t(		**t(**
NUM	Abs		**Abs_**
	Int		**Int_**
	Frac		**Frac_**
	Rnd		**Rnd**
	Intg		**Intg_**
	RndFi		**RndFix(**
	GCD		**GCD(**
	LCM		**LCM(**
	MOD		**MOD(**
	MOD•E		**MOD_Exp(**
ANGL	°		°
	r		**r**
	g		**g**
	° ’ ”		◘
	Pol(		**Pol(**
	Rec(		**Rec(**
	▶DMS		**▶DMS**
ESYM	m		**m**
	μ		μ
	n		**n**
	p		**p**
	f		**f**
	k		**k**
	M		**M**
	G		**G**
	T		**T**
	P		**P**
	E		**E**
PICT	Sto		**StoPict_**
	Rcl		**RclPict_**
FMEM	fn		**fn**
LOGIC	And		**_And_**
	Or		**_Or_**
	Not		**Not_**
	Xor		**Xor_**
CAPT	Rcl		**RclCapt_**
TVM	SMPL	SI	**Smpl_SI(**
		SFV	**Smpl_SFV(**
	CMPD	n	**Cmpd_n(**
		I%	**Cmpd_I%(**
		PV	**Cmpd_PV(**
		PMT	**Cmpd_PMT(**
		FV	**Cmpd_FV(**
	CASH	NPV	**Cash_NPV(**
		IRR	**Cash_IRR(**
		PBP	**Cash_PBP(**
		NFV	**Cash_NFV(**
	AMT	BAL	**Amt_BAL(**
		INT	**Amt_INT(**
		PRN	**Amt_PRN(**
		ΣINT	**Amt_ΣINT(**
		ΣPRN	**Amt_ΣPRN(**
	CNVT	EFF	**Cnvt_EFF(**
		APR	**Cnvt_APR(**
	COST	Cost	**Cost(**

		Sell	**Sell(**
		Mrg	**Margin(**
	DAYS	PRD	**Days_Prd(**
	BOND	PRC	**Bond_PRC(**
		YLD	**Bond_YLD(**

VARS key

Level 1	Level 2	Level 3	Command
V-WIN	X	min	**Xmin**
		max	**Xmax**
		scal	**Xscl**
		dot	**Xdot**
	Y	min	**Ymin**
		max	**Ymax**
		scal	**Yscl**
	T,θ	min	**Tθmin**
		max	**Tθmax**
		ptch	**Tθptch**
	R-X	min	**RightXmin**
		max	**RightXmax**
		scal	**RightXscl**
		dot	**RightXdot**
	R-Y	min	**RightYmin**
		max	**RightYmax**
		scal	**RightYscl**
	R-T, θ	min	**RightTθmin**
		max	**RightTθmax**
		ptch	**RightTθptch**
FACT	Xfct		**Xfct**
	Yfct		**Yfct**
STAT	X	n	**n**
		$\bar{x}$	$\bar{x}$
		Σx	**Σx**
		Σx^2	Σx^2
		σ_x	σ_x
		s_x	s_x
		minX	**minX**
		maxX	**maxX**
	Y	$\bar{y}$	$\bar{y}$
		Σy	**Σy**
		Σy^2	Σy^2
		Σxy	**Σxy**
		σ_y	σ_y
		s_y	s_y
		minY	**minY**
		maxY	**maxY**
	GRPH	a	**a**
		b	**b**
		c	**c**
		d	**d**
		e	**e**
		r	**r**
		r^2	r^2
		MSe	**MSe**
		Q_1	Q_1
		Med	**Med**
		Q_3	Q_3
		Mod	**Mod**
		Strt	**H_Start**
		Pitch	**H_pitch**
	PTS	x_1	x_1
		y_1	y_1
		x_2	x_2
		y_2	y_2
		x_3	x_3
		y_3	y_3
	INPT	n	**n**
		$\bar{x}$	$\bar{x}$
		s_x	s_x
		n1	**n1**
		n2	**n2**
		$\bar{x}_1$	$\bar{x}_1$
		$\bar{x}_2$	$\bar{x}_2$
		s_{x1}	s_{x1}
		s_{x2}	s_{x2}
		s_p	s_p
	RESLT	*7	
GRPH	Y		**Y**
	r		**r**
	Xt		**Xt**
	Yt		**Yt**
	X		**X**
DYNA	Strt		**D_Start**
	End		**D_End**
	Pitch		**D_pitch**
TABL	Strt		**F_Start**
	End		**F_End**
	Pitch		**F_pitch**
	Reslt		**F_Result**
RECR	FORM	a_n	a_n
		a_{n+1}	a_{n+1}
		a_{n+2}	a_{n+2}
		b_n	b_n
		b_{n+1}	b_{n+1}
		b_{n+2}	b_{n+2}
		c_n	c_n
		c_{n+1}	c_{n+1}
		c_{n+2}	c_{n+2}
	RANG	Strt	**R_Start**
		End	**R_End**
		a_0	a_0
		a_1	a_1
		a_2	a_2
		b_0	b_0
		b_1	b_1
		b_2	b_2
		c_0	c_0
		c_1	c_1
		c_2	c_2
		a_nSt	a_n**Start**
		b_nSt	b_n**Start**
		c_nSt	c_n**Start**
	Reslt		**R_Result**
EQUA	S-Rlt		**Sim_Result**
	S-Cof		**Sim_Coef**
	P-Rlt		**Ply_Result**
	P-Cof		**Ply_Coef**
TVM	n		**n**
	I%		**I%**

	PV		**PV**
	PMT		**PMT**
	FV		**FV**
	P/Y		**P/Y**
	C/Y		**C/Y**
Str			**Str_**

SHIFT VARS (PRGM) key

Level 1	Level 2	Level 3	Command
COM	If		**If_**
	Then		**Then_**
	Else		**Else_**
	I-End		**IfEnd**
	For		**For_**
	To		**_To_**
	Step		**_Step_**
	Next		**Next**
	Whle		**While_**
	WEnd		**WhileEnd**
	Do		**Do**
	Lp-W		**LpWhile_**
CTL	Prog		**Prog_**
	Rtrn		**Return**
	Brk		**Break**
	Stop		**Stop**
JUMP	Lbl		**Lbl_**
	Goto		**Goto_**
	⇒		**⇒**
	Isz		**Isz_**
	Dsz		**Dsz_**
	Menu		**Menu_**
?			**?**
◢			**◢**
CLR	Text		**ClrText**
	Grph		**ClrGraph**
	List		**ClrList_**
	Mat		**ClrMat_**
	Vct		**ClrVct_**
DISP	Stat		**DrawStat**
	Grph		**DrawGraph**
	Dyna		**DrawDyna**
	F-Tbl	Tabl	**DispF-Tbl**
		G-Con	**DrawFTG-Con**
		G-Plt	**DrawFTG-Plt**
	R-Tbl	Tabl	**DispR-Tbl**
		Phase	**PlotPhase**
		Web	**DrawWeb_**
		an-Cn	**DrawR-Con**
		Σa-Cn	**DrawR Σ-Con**
		an-Pl	**DrawR-Plt**
		Σa-Pl	**DrawR Σ-Plt**
REL	=		**=**
	≠		**≠**
	>		**>**
	<		**<**
	≥		**≥**
	≤		**≤**
I/O	Lcte		**Locate_**
	Gtky		**Getkey**
	Send		**Send(**
	Recv		**Receive(**
	S38k		**Send38k_**
	R38k		**Receive38k_**
	Open		**OpenComport38k**
	Close		**CloseComport38k**
:			**:**
STR	Join		**StrJoin(**
	Len		**StrLen(**
	Cmp		**StrCmp(**
	Src		**StrSrc(**
	Left		**StrLeft(**
	Right		**StrRight(**
	Mid		**StrMid(**
	E▶S		**Exp▶Str(**
	Exp		**Exp(**
	Upr		**StrUpr(**
	Lwr		**StrLwr(**
	Inv		**StrInv(**
	Shift		**StrShift(**
	Rot		**StrRotate(**

SHIFT MENU (SET UP) key

Level 1	Level 2	Level 3	Command
ANGL	Deg		**Deg**
	Rad		**Rad**
	Gra		**Gra**
COOR	On		**CoordOn**
	Off		**CoordOff**
GRID	On		**GridOn**
	Off		**GridOff**
AXES	On		**AxesOn**
	Off		**AxesOff**
LABL	On		**LabelOn**
	Off		**LabelOff**
DISP	Fix		**Fix_**
	Sci		**Sci_**
	Norm		**Norm_**
	Eng	On	**EngOn**
		Off	**EngOff**
		Eng	**Eng**
S/L	—		**S-L-Normal**
	—		**S-L-Thick**
	·····		**S-L-Broken**
	······		**S-L-Dot**
DRAW	Con		**G-Connect**
	Plot		**G-Plot**
DERV	On		**DerivOn**
	Off		**DerivOff**
BACK	None		**BG-None**
	Pict		**BG-Pict_**
FUNC	On		**FuncOn**
	Off		**FuncOff**
SIML	On		**SimulOn**
	Off		**SimulOff**
S-WIN	Auto		**S-WindAuto**
	Man		**S-WindMan**
LIST	File		**File_**
LOCS	On		**LocusOn**
	Off		**LocusOff**
T-VAR	Rang		**VarRange**
	List		**VarList_**
ΣDSP	On		**ΣdispOn**
	Off		**ΣdispOff**
RESID	None		**Resid-None**
	List		**Resid-List_**
CPLX	Real		**Real**
	a+bi		**a+bi**
	r∠θ		**r∠θ**
FRAC	d/c		**d/c**
	ab/c		**ab/c**
Y•SPD	Norm		**Y=DrawSpeedNorm**
	High		**Y=DrawSpeedHigh**
DATE	365		**DateMode365**
	360		**DateMode360**
PMT	Bgn		**PmtBgn**
	End		**PmtEnd**
PRD	Annu		**PeriodsAnnual**
	Semi		**PeriodsSemi**
INEQ	And		**IneqTypeAnd**
	Or		**IneqTypeOr**
SIMP	Auto		**SimplfyAuto**
	Man		**SimplfyMan**
Q1Q3	Std		**Q1Q3TypeStd**
	OnD		**Q1Q3TypeOnData**

SHIFT key

Level 1	Level 2	Level 3	Command
ZOOM	Fact		**Factor_**
	Auto		**ZoomAuto**
V-WIN	V-Win		**ViewWindow_**
	Sto		**StoV-Win_**
	Rcl		**RclV-Win_**
SKTCH	Cls		**Cls**
	Tang		**Tangent_**
	Norm		**Normal_**
	Inv		**Inverse_**
	GRPH	Y=	**Graph_Y=**
		r=	**Graph_r=**
		Parm	**Graph(X,Y)=(**
		X=c	**Graph_X=**
		G-∫dx	**Graph_∫**
		Y>	**Graph_Y>**
		Y<	**Graph_Y<**
		Y≥	**Graph_Y≥**
		Y≤	**Graph_Y≤**
		X>	**Graph_X>**
		X<	**Graph_X<**
		X≥	**Graph_X≥**
		X≤	**Graph_X≤**
	PLOT	Plot	**Plot_**
		Pl-On	**PlotOn_**
		Pl-Off	**PlotOff_**
		Pl-Chg	**PlotChg_**
	LINE	Line	**Line**
		F-Line	**F-Line_**
	Crcl		**Circle_**
	Vert		**Vertical_**

	Hztl		Horizontal_
	Text		Text_
	PIXL	On	PxlOn_
		Off	PxlOff_
		Chg	PxlChg_
	Test		PxlTest(
	STYL	—	SketchNormal_
		━	SketchThick_
		·····	SketchBroken_
		······	SketchDot_

BASE Program

F4 (MENU) key

Level 1	Level 2	Level 3	Command
d~o	d		d
	h		h
	b		b
	o		o
LOG	Neg		Neg_
	Not		Not_
	and		and
	or		or
	xor		xor
	xnor		xnor
DISP	▶Dec		▶Dec
	▶Hex		▶Hex
	▶Bin		▶Bin
	▶Oct		▶Oct

SHIFT VARS (PRGM) key

Level 1	Level 2	Level 3	Command
Prog			Prog_
JUMP	Lbl		Lbl_
	Goto		Goto_
	⇒		⇒
	Isz		Isz_
	Dsz		Dsz_
	Menu		Menu_
?			?
◢			◢
REL	=		=
	≠		≠
	>		>
	<		<
	≥		≥
	≤		≤
:			:

SHIFT MENU (SET UP) key

Level 1	Level 2	Level 3	Command
Dec			Dec
Hex			Hex
Bin			Bin
Oct			Oct

	Level 3	Level 4	Command
*1	Exp	ae^bx	Exp(ae^bx)
		ab^x	Exp(ab^x)
*2	MARK	◘	Square
		×	Cross
		■	Dot
	STICK	Leng	StickLength
		Hztl	StickHoriz
	%DATA	%	%
		Data	Data
	None		None
*3	X	ax+b	LinearReg(ax+b)
		a+bx	LinearReg(a+bx)
*4	EXP	ae^bx	ExpReg(a•e^bx)
		ab^x	ExpReg(a•b^x)
*5	NORM	NPd	NormPD(
		NCd	NormCD(
		InvN	InvNormCD(
	t	TPd	tPD(
		TCd	tCD(
		Invt	InvTCD(
	CHI	CPd	ChiPD(
		CCd	ChiCD(
		InvC	InvChiCD(
	F	FPd	FPD(
		FCd	FCD(
		InvF	InvFCD(
	BINM	BPd	BinomialPD(
		BCd	BinomialCD(
		InvB	InvBinomialCD(
	POISN	PPd	PoissonPD(
		PCd	PoissonCD(
		InvP	InvPoissonCD(
	GEO	GPd	GeoPD(
		GCd	GeoCD(
		InvG	InvGeoCD(
	H•GEO	HPd	HypergeoPD(
		HCd	HypergeoCD(
		InvH	InvHyperGeoCD(
*6	Z	1-S	OneSampleZTest_
		2-S	TwoSampleZTest_
		1-P	OnePropZTest_
		2-P	TwoPropZTest_
	t	1-S	OneSampleTTest_
		2-S	TwoSampleTTest_
		REG	LinRegTTest_
	Chi	GOF	ChiGOFTest_
		2-WAY	ChiTest_
	F		TwoSampleFTest_
	ANOV	1-W	OneWayANOVA_
		2-W	TwoWayANOVA_
*7	TEST	p	p
		z	z
		t	t
		Chi	χ^2
		F	F
		$\hat{p}$	$\hat{p}$
		$\hat{p}_1$	$\hat{p}_1$
		$\hat{p}_2$	$\hat{p}_2$
		df	df

		Se	Se
		r	r
		r^2	r^2
		pa	pa
		Fa	Fa
		Adf	Adf
		SSa	SSa
		MSa	MSa
		pb	pb
		Fb	Fb
		Bdf	Bdf
		SSb	SSb
		MSb	MSb
		pab	pab
		Fab	Fab
		ABdf	ABdf
		SSab	SSab
		MSab	MSab
		Edf	Edf
		SSe	SSe
		MSe	MSe
	INTR	Left	Left
		Right	Right
		$\hat{p}$	$\hat{p}$
		$\hat{p}_1$	$\hat{p}_1$
		$\hat{p}_2$	$\hat{p}_2$
		df	df
	DIST	p	p
		xInv	xInv
		x1Inv	x1Inv
		x2Inv	x2Inv
		zLow	zLow
		zUp	zUp
		tLow	tLow
		tUp	tUp

8. Program Library

- Be sure to check how many bytes of unused memory are remaining before attempting to perform any programming.

Program Name	**Prime Factorization**

Description

This program continually divides a natural number by factors until all its prime factors are produced.

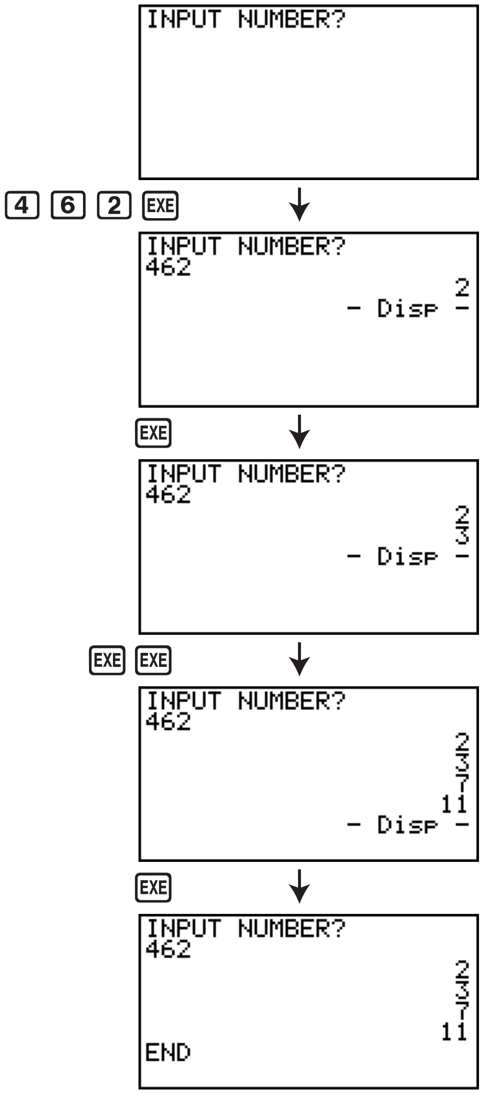
Purpose

This program accepts input of natural number A, and divides it by B (2, 3, 5, 7....) to find the prime factors of A.

- If a division operation does not produce a remainder, the result of the operation is assigned to A.
- The above procedure is repeated until B > A.

Example **462 = 2 × 3 × 7 × 11**

```
ClrText↵
"INPUT NUMBER"?→A↵
2→B↵
Do↵
While Frac (A÷B)=0↵
B◢
A÷B→A↵
WhileEnd↵
If B=2↵
Then 3→B↵
Else B+2→B↵
IfEnd↵
LpWhile B≤A↵
"END"
```



```
INPUT NUMBER?
```

4 6 2 EXE

```
INPUT NUMBER?
462
                   2
          - Disp -
```

EXE

```
INPUT NUMBER?
462
                   2
                   3
          - Disp -
```

EXE EXE

```
INPUT NUMBER?
462
                   2
                   3
                   7
                  11
          - Disp -
```

EXE

```
INPUT NUMBER?
462
                   2
                   3
                   7
                  11
END
```

Program Name Ellipse

Description

This program displays a number table of the following values based on input of the foci of an ellipse, the sum of the distance between the loci and foci, and the pitch (step size) of X.

Y1: Coordinate values of upper half of ellipse

Y2: Coordinate values of lower half of ellipse

Y3: Distances between right focus and loci

Y4: Distances between left focus and loci

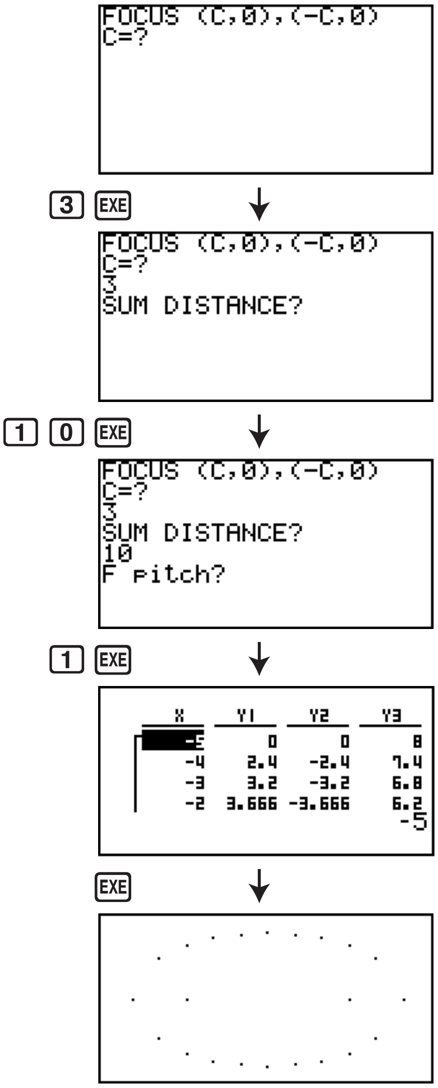
Y5: Sum of Y3 and Y4

Next, the program plots the foci and values in Y1 and Y2.

Purpose

This program shows that the sums of the distances between the loci and two foci of an ellipse are equal.

```
AxesOff↵
Do↵
ClrText↵
"FOCUS (C,0),(-C,0)"↵
"C="?→C↵
"SUM DISTANCE"?→D↵
LpWhile 2Abs C≥D Or D≤0↵
D÷2→A↵
√(A²−C²)→B↵
Y=Type↵
"B√(1−X²÷A²)"→Y1↵
"−Y1"→Y2↵
"√((X−C)²+Y1²)"→Y3↵
"√((X+C)²+Y1²)"→Y4↵
"Y3+Y4"→Y5↵
For 1→E To 20↵
If E≤5↵
Then T SelOn E↵
Else T SelOff E↵
IfEnd↵
Next↵
−Int A→F Start↵
Int A→F End↵
"F pitch"?→F pitch↵
DispF-Tbl◢
ClrGraph↵
1.2A→Xmax↵
−1.2A→Xmin↵
1.2B→Ymax↵
−1.2B→Ymin↵
T SelOff 3↵
T SelOff 4↵
T SelOff 5↵
DispF-Tbl↵
DrawFTG-Plt↵
PlotOn C,0↵
PlotOn −C,0◢
"END"
```



```
FOCUS (C,0),(-C,0)
C=?
```

```
FOCUS (C,0),(-C,0)
C=?
3
SUM DISTANCE?
```

```
FOCUS (C,0),(-C,0)
C=?
3
SUM DISTANCE?
10
F pitch?
```

```
      X      Y1      Y2     Y3
     -5       0       0      8
     -4     2.4    -2.4    7.4
     -3     3.2    -3.2    6.8
     -2   3.666  -3.666    6.2
                            -5
```